HP StorageWorks Data Exchange XP user guide

Product Version: 2.05.00

fourth edition (April 2005)

part number: B7914-96003

This guide explains how to use HP StorageWorks Data Exchange XP software.

Printed in the U.S.A.

HP StorageWorks Data Exchange XP User Guide

fourth edition (April 2005)
part number: B7914-96003

Contents

About this guide

This guide explains how to use HP StorageWorks Data Exchange XP, a program designed to allow seamless data exchange between mainframe and open system hosts. It manages data format and code conversions and allows information sharing across computing platforms.

Intended audience

This guide is intended for system managers who have knowledge of:

- Data processing concepts
- Disk array management and basic functions
- Operating system commands and utilities
- Mainframe operations

Disk arrays

Unless otherwise noted, the term *disk array* refers to these disk arrays:

HP Surestore Disk Array XP512
HP Surestore Disk Array XP48
HP StorageWorks Disk Array XP128
HP StorageWorks Disk Array XP1024
HP StorageWorks XP12000 Disk Array

Related documentation

HP provides these related documents:

- *HP StorageWorks Disk Array XP128: Owner's Guide*
- *HP StorageWorks Disk Array XP1024: Owner's Guide*
- *HP StorageWorks XP12000 Disk Array: Owner's Guide*

Refer to the manufacturer's documentation for information about operating system commands and third-party products.

Conventions

This guide uses these text conventions.

Figure 1	Blue text represents a cross-reference. In the online version of this guide, the reference is linked to the target.
www.hp.com	Underlined, blue text represents a website on the Internet. In the online version of this guide, the reference is linked to the target.
literal	Bold text represents application names, file names, menu items, dialog box titles, buttons, key names, field names, and literal values that you type exactly as shown.
variable	Italic type indicates that you must supply a value. Italic type is also used for manual titles.
`input/output`	Monospace font denotes user input and system responses, such as output and messages.
Example	The word “example” in italics denotes an example of input or output.
[ ]	Square brackets indicate an optional parameter.
{ }	Braces indicate that you must specify at least one of the listed options.
|	A vertical bar separates alternatives in a list of options.

HP technical support

In North America, call technical support at 1-800-652-6672, available 24 hours a day, 7 days a week.

Outside North America, call technical support at the nearest location. Telephone numbers for worldwide technical support are listed on the HP website under support:

http://h18006.www1.hp.com/storage/arraysystems.html

Be sure to have the following information available before calling:

- Technical support registration number (if applicable)
- Product serial numbers
- Product model names and numbers
- Applicable error messages
- Operating system type and revision level
- Detailed, specific questions

For continuous quality improvement, calls may be recorded or monitored.

HP storage website

Visit the support website for the most current information about HP StorageWorks XP products.

http://h18006.www1.hp.com/storage/arraysystems.html

Consult your HP account representative for information about product availability, configuration, and connectivity.

HP authorized reseller

For the name of your nearest HP authorized reseller, call:

United States	1-800-345-1518
Canada	1-800-263-5868
Or contact:	www.hp.com

Revision history

June 1999	First edition.
September 1999	Second edition.
July 2002	Third edition
April 2005	Fourth edition

Warranty statement

HP warrants that for a period of ninety calendar days from the date of purchase, as evidenced by a copy of the invoice, the media on which the Software is furnished (if any) will be free of defects in materials and workmanship under normal use.

DISCLAIMER. EXCEPT FOR THE FOREGOING AND TO THE EXTENT ALLOWED BY LOCAL LAW, THIS SOFTWARE IS PROVIDED TO YOU “AS IS” WITHOUT WARRANTIES OF ANY KIND, WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED. HP SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT, TITLE, ACCURACY OF INFORMATIONAL CONTENT, AND FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow exclusions of implied warranties or conditions, so the above exclusion may not apply to you to the extent prohibited by such local laws. You may have other rights that vary from country to country, state to state, or province to province.

WARNING! **YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT USE OF THE SOFTWARE IS AT YOUR SOLE RISK.** HP DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED, VIRUS-FREE OR ERROR-FREE, OR THAT DEFECTS IN THE SOFTWARE WILL BE CORRECTED. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY YOU. HP DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE OR RELATED DOCUMENTATION IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS, OR OTHERWISE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY HP OR HP’S AUTHORIZED REPRESENTATIVES SHALL CREATE A WARRANTY.

LIMITATION OF LIABILITY. EXCEPT TO THE EXTENT PROHIBITED BY LOCAL LAW, IN NO EVENT INCLUDING NEGLIGENCE WILL HP OR ITS SUBSIDIARIES, AFFILIATES, DIRECTORS, OFFICERS, EMPLOYEES, AGENTS OR SUPPLIERS BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR OTHER DAMAGES (INCLUDING LOST PROFIT, LOST DATA, OR DOWNTIME COSTS), ARISING OUT OF THE USE, INABILITY TO USE, OR THE RESULTS OF USE OF THE SOFTWARE, WHETHER BASED IN WARRANTY, CONTRACT, TORT OR OTHER LEGAL THEORY, AND WHETHER OR NOT ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Your use of the Software is entirely at your own risk. Should the Software prove defective, you assume the entire cost of all service, repair or correction. Some jurisdictions do not allow the exclusion or limitation of liability for incidental or consequential damages, so the above limitation may not apply to you to the extent prohibited by such local laws.

NOTE. EXCEPT TO THE EXTENT ALLOWED BY LOCAL LAW, THESE WARRANTY TERMS DO NOT EXCLUDE, RESTRICT OR MODIFY, AND ARE IN ADDITION TO, THE MANDATORY STATUTORY RIGHTS APPLICABLE TO THE LICENSE OF THE SOFTWARE TO YOU; PROVIDED, HOWEVER, THAT THE CONVENTION ON CONTRACTS FOR THE INTERNATIONAL SALE OF GOODS IS SPECIFICALLY DISCLAIMED AND SHALL NOT GOVERN OR APPLY TO THE SOFTWARE PROVIDED IN CONNECTION WITH THIS WARRANTY STATEMENT.

1 Overview

Data Exchange (DE) converts and transfers data between different platforms.

Data transfers types are:

- **Mainframe to open system (MTO) -** Transfer data from S/390 (mainframe) datasets to open system files.
- **Open system to mainframe (OTM) -** Transfer data from open system files to S/390 datasets.
- **Open system to open system (OTO) -** Transfer data between open systems.

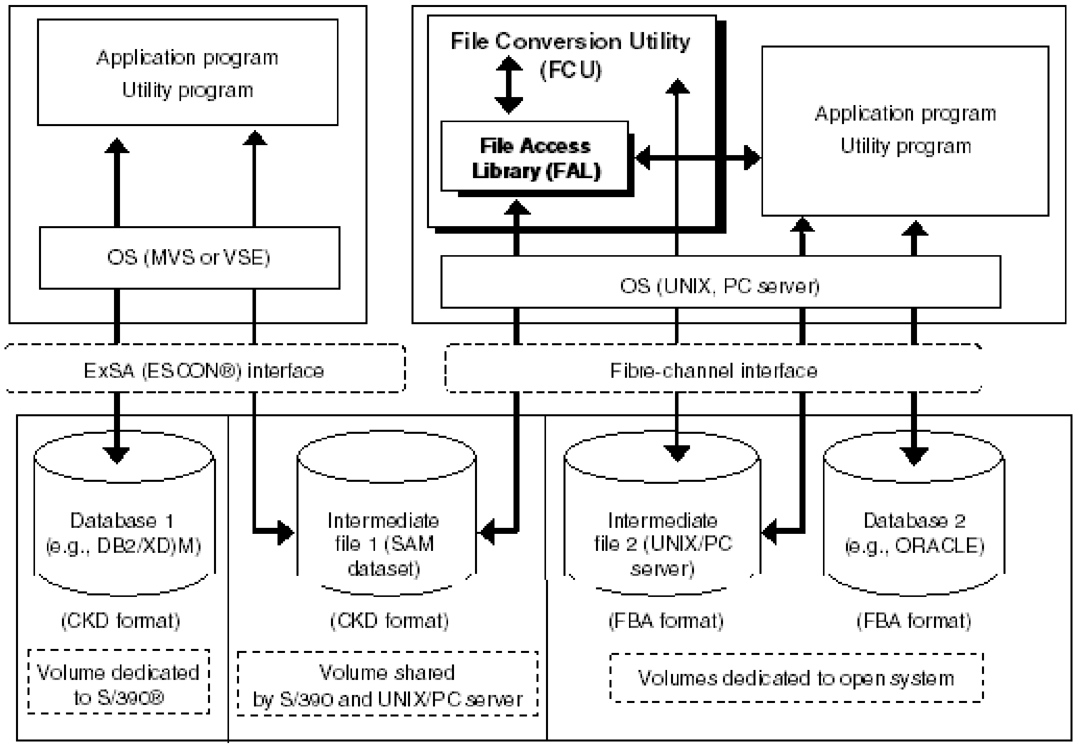
How Data Exchange works

Data Exchange consists of the following:

- **File Conversion Utility (FCU).** FCU provides the GUI and commands for file transfer operations. It includes data exchange options, including EBCDIC-ASCII code conversion and data record padding and delimiters.
- **File Access Library (FAL).** FAL is a library of C functions (Visual C++ for Windows) that provides an application programming interface (API) for data exchange. FAL functions can be called by application programs to read and write data in S/390 datasets on the disk array. There are two types of FAL, 32-bit and 64-bit. The XP128 uses 32-bit FAL and the XP1024 and XP12000 use 64-bit FAL.
- **Formatter (FMT) and Allocator (ALC).** The FMT and ALC utilities are used to format OPEN-*x* logical units (LUNs) and create intermediate datasets for OTO operations. See "Formatting OTO volumes using the FMT utility" (page 39) and "Allocating OTO intermediate datasets using ALC" (page 90).

MTO and OTM operations

This illustration provides an overview of the data exchange process between mainframes and open systems (MTO and OTM transfers).


Application program
Utility program
OS (MVS or VSE)
File Conversion Utility (FCU)
File Access Library (FAL)
Application program
Utility program
OS (UNIX, PC server)
ExSA (ESCON®) interface
Fibre-channel interface
Database 1 (e.g., DB2/XD)M)
(CKD format)
Volume dedicated to S/390®
Intermediate file 1 (SAM dataset)
(CKD format)
Volume shared by S/390 and UNIX/PC server
Intermediate file 2 (UNIX/PC server)
(FBA format)
Database 2 (e.g., ORACLE)
(FBA format)
Volumes dedicated to open system

MTO and OTM operations are performed using the File Conversion Utility and File Access Library, which are installed on the open system host.

Data Exchange uses special volumes that are dedicated to data exchange operations. These are accessed as raw devices to provide multiplatform data exchange.

Data Exchange supports MVS and VSE mainframe operating systems. Please note the following:

VSE

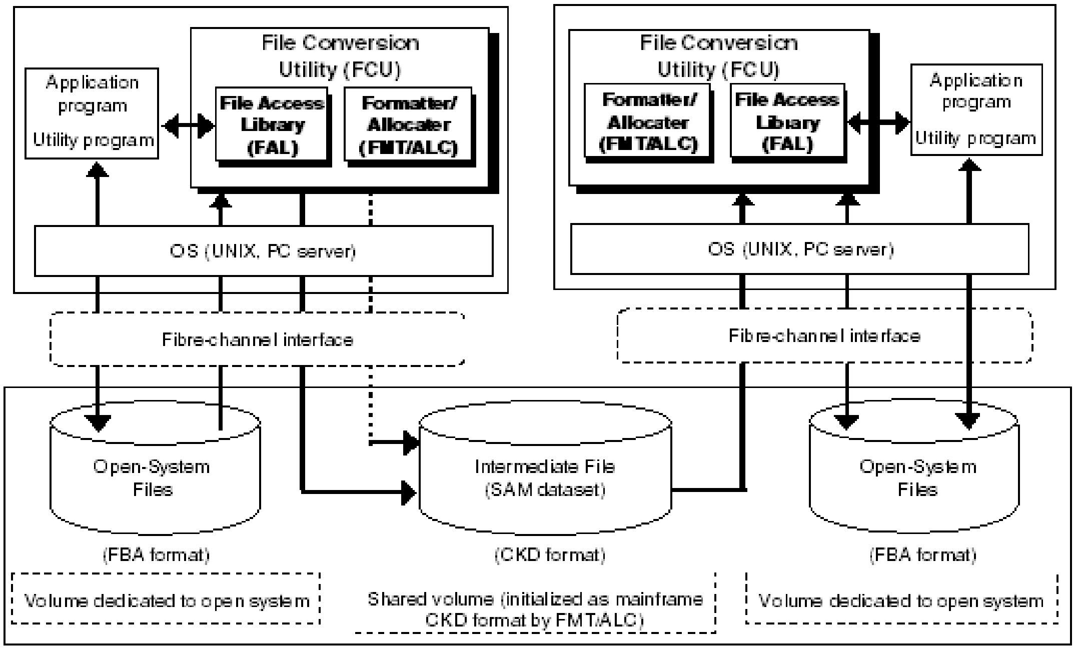
- Only versions 2.3 and earlier of VSE are supported.
- Multiple Volume Dataset is not supported.

MVS

- Multiple Volume Dataset is supported for MTO operations only.

OTO operations

This illustration provides an overview of the data exchange process between two open systems (OTO transfers).

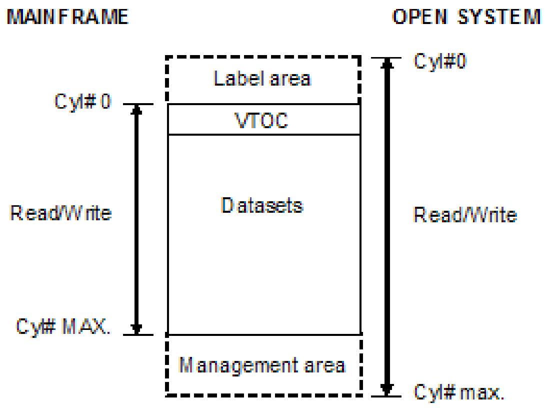

File Conversion Utility (FCU)
Application program
Utility program
File Access Library (FAL)
Formatter/ Allocater (FMT/ALC)
OS (UNIX, PC server)
Fibre-channel interface
File Conversion Utility (FCU)
Formatter/ Allocater (FMT/ALC)
File Access Library (FAL)
Application program
Utility program
OS (UNIX, PC server)
Fibre-channel interface
Open-System Files
(FBA format)
Volume dedicated to open system
Intermediate File (SAM dataset)
(CKD format)
Shared volume (initialized as mainframe CKD format by FMT/ALC)
Open-System Files
(FBA format)
Volume dedicated to open system

OTO operations are performed using the Formatter and Allocater utilities in addition to FCU and FAL.

Data Exchange volume types

Data Exchange operations are performed using the following types of Data Exchange volumes:

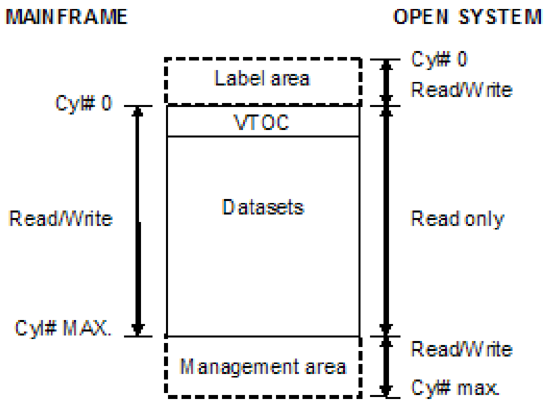
- **-A volumes.** -A volumescan be used for MTO and OTM operations.



S/390 hosts have normal read/write access to -A volumesOpen system hosts have read/write access to -A volumes but must use DE to access these volumes as raw devices (no mount operation).

Note: The -A volumes are not write-protected. Do not execute any open system write operations to -A volumes (except disk partitioning and labeling). Do not create a file system on an -A volume; this will overwrite the data exchange files on the volume.

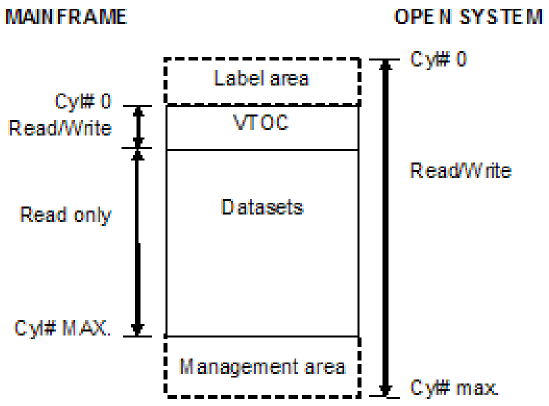
- **-B volumes. -**B volumescan only be used for MTO operations.



S/390 hosts have normal read/write access to -B volumesOpen system hosts have read-only access to -B volumes and must use DE to read these volumes as raw devices (no mount operation). The -B volumes are write-protected from open system access.The disk arrays will reject all open system write operations to -B volumes (except disk partitioning and labeling) to protect the S/390 data on these volumes.

Note: The open system host accesses only the volume table of contents (VTOC) area on -B volumes. Catalog or security control functions cannot be used to provide access control for these volumes.

- **-C volumes.** -C volumescan only be used for OTM operations.



Open system hosts have read/write access to the -C volumes but must use DE to access these volumes as raw devices (no mount operation).

S/390 hosts have read-only access to the -C volumes. The disk arrays will reject all S/390 write operations to -C volumes (except VTOC) to protect the open system data on these volumes.

- **OPEN-*x*.** OPEN-*x* volumes can only be used for OTO operations and must be formatted using the FMT utility.

  Open system hosts have read/write access to the OPEN-*x* FMT volumes but must use FAL/FCU to access these volumes as raw devices (no mount operation).

  S/390 hosts do not have any access to the OPEN-*x* FMT volumes.

  Note: For VSE mainframe systems, DFSORT cannot be used after an Data Exchange volume is copied to another volume. Use the DITTO function for this purpose.

Notes for Microsoft Cluster Server

When installing DE devices in a Microsoft Cluster Server (MSCS) environment, you must write signatures on the DE volumes before configuring MSCS. Note:

- The MSCS server cannot connect volumes that do not have signatures.
- The volume on which a signature is written cannot be accessed from another server.
- The volume on which a signature is written cannot be shared.
- Only the mainframe and the server that wrote the signature can access the volume that has the signature.
- Signatures cannot be written to DE volumes for which the emulation type is 3390-3X/9X/LX or 3380-KX (X = A, B, C), when the OS server is Windows 2000/2003/NT (3390-3X for XP128, XP1024, and XP12000 subsystems; 3390-9X/LX for XP1024 and XP12000 only; 3380-KX for only).
- When configuring MSCS and the server OS is Windows 2000/2003/NT, OTM and MTO cannot be started.

Note for Windows

When installing DE in a Windows NT/2000/2003 environment, note:

- Service Pack 1 must be installed when MSCS is configured.

- A signature is not necessary for the MSCS configuration. A write error will occur if a signature is attempted.

Caution *Do not write a signature on DE volumes having emulation types in a Windows NT/2000/2003 environment. If a signature-writing attempt is made by the Disk Administrator with Windows NT/2000/2003, a Write Error will appear in order to stop the signature from being written. When the Windows NT/2000/2003 Disk Administrator starts again, a request will be made again to write the signature. Do not write the signature.*

Maximum data size

The following table provides the maximum data size for DE for different emulation types.

Note: The data capacity that can be stored within the intermediate file is smaller than its physical capacity, and varies depending upon the block length to be used.

2

Installation

This chapter explains how to install and configure Data Exchange.

It is important to determine the how many volumes are needed for Data Exchange prior to configuring your disk array. Data Exchange volumes should be set up when you first configure your disk array, as reconfiguring DE volumes after configuring the disk array may require reformatting entire array groups, depending on the microcode level of the XP series.

System requirements

The system requirements for DE are:

- Multiplatform or all-open StorageWorks XP disk arrays, unconfigured.

Caution *Data Exchange volumes should be set up when you first configure your disk array, as reconfiguring DE volumes after configuring the disk array may require reformatting entire array groups. See "Installing and configuring the DE volumes" (page 26) for more information.*

- HP StorageWorks Remote Control XP and HP StorageWorks LUN Manager XP software. LUN Manager is used to configure FC ports and create custom-size LUNs.
- S/390 operating systems: MVS, VSE, or .
- Open system platforms and operating system (OS) version level:

  32-bit FAL

  - HP-UX 10.2 and 11.0
  - Sun Solaris 2.5, 2.6, 7, 8, 9 (version 2.5 = Ver. 01-XX-47 or earlier)
  - IBM AIX 5.2, 5.1, 4.1.x, 4.2.x, 4.3.x, 5 (version 4.1 = Ver. 01-XX-47 or earlier)
  - Windows 2000/2003/NT 4.0/IA64 (Workstation or Server)
  - Compaq Tru64 UNIX:Digital UNIX 4.0; Tru64 UNIX 4.0, 5.0
  - Red Hat Linux 7.2: AS2.1, AS3.0/IA64

  64-bit FAL

  - HP-UX 11.0
  - Sun Solaris 7, 8, 9
  - IBM AIX 5.2, 5.1, 4.3.3
  - Red Hat Linux AS3.0/IA64

64-bit FCU

- HP-UX 11.0,11i,11iV2
- Sun Solaris 7,8,9
- IBM AIX 4.3, 5L
- Red Hat Linux AS3.0/IA64

- Motif 1.2 (or later) window system software is required for the FCU GUI for UNIX. If Motif is not installed, see Appendix A, "Using FCU without the GUI" (page 195)
- Superuser (root) login access to the open system server/workstation is required.

Installing and configuring the DE volumes

The Data Exchange volumes should be installed and configured during initial installation and configuration of the disk array. Creating DE volumes after the disk array is configured may result in the need to reformat entire arrays.

The DE volumes should be dedicated to data exchange operations to avoid accidental overwriting or deletion of important data. Note:

- **MTO**. For mainframe to open system operations, the disk array must be configured with **-B** and/or **-A** volumes dedicated to DE. MTO volumes contain S/390 data to be transferred to open system LUNs.
- **OTM**. For open system to mainframe operations, the disk array must be configured with **-C** and/or **-A** volumes dedicated to DE. OTM volumes contain open system data to be transferred to S/390 volumes.
- **OTO.** For open system to open system operations, the disk array must be configured with **OPEN-*x* LUNs** as OTO volumes or **-C** volumes dedicated to DE. OTO volumes (-C or OPEN-*x* LUNs) contain the intermediate datasets for file transfers between open system platforms.

DE does not support concurrent access to DE volumes by the S/390 and open system hosts.

Determine DE volumes needed

Determine exactly how many MTO, OTM, and OTO volumes you will need for your multi platform data exchange operations. Note:

- The -A volumes can be used for MTO, OTM, and OTO.
- The -B volumes are restricted to MTO.
- The -C volumes are restricted to OTM.
- The OPEN-*x* FMT volumes are restricted to OTO.

Make sure that the HP representative installs the desired number of each type of DE volume during disk array installation and configuration.

Note: If you need to change the number of DE volumes, contact your HP representative. Reconfiguring the DE volumes after disk array installation may require reformatting entire array groups.

Install and configure disk array

The next step is to complete disk array installation and device configuration as specified in the HP OS configuration guide for the applicable open system platform.

This section highlights important points that are necessary when configuring your disk array to work with Data Exchange.

Device recognition and device files

Verify device recognition and device file creation for all DE volumes.

File system/volume group

Do not create a file system or volume group on any DE volume, including the OPEN-*x* devices which will be formatted for OTO operations. DE volumes can only be accessed as raw devices by the open system host using DE (no mount operation).

Defining RAW devices

Those volumes which are to be used as intermediate volumes and to be shared between open systems for OTO operations must be defined as **OPEN-3/8/9/K/E/L/M/V** emulation type for RAID200/300 and must be defined as “raw” devices from each host server. From the open systems, there are no means to distinguish OPEN-3/8/9/K/E/L/M/V for open system dedicated volumes from these DE volumes. Please make sure not to confuse the usage on those volumes in the host systems.

The operations below which create file systems on the intermediate volumes must not be executed. Otherwise, information on the volume may be destroyed and the volumes will become unusable as DE volumes.

- SUN Solaris: “**newfs**” command

- HP-UX: “**pvcreate**” command
- IBM AIX: creating a volume group
- Windows: Formatting and creating a file system
- Digital UNIX/HP Tru64 UNIX: “**newfs**” command
- Sequent Dynix/ptx: creating a file system
- NCR SVR4: creating a volume group
- Linux: “**raw**” command

Note for Microsoft Cluster Server:

When installing DE devices in a Microsoft Cluster Server (MSCS) environment, you must write signatures on the DE volumes before configuring MSCS. Note:

- The MSCS server cannot connect volumes which do not have signatures.
- The volume on which a signature is written cannot be accessed from another server.
- The volume on which a signature is written cannot be shared.
- Only the mainframe and the server which wrote the signature can access the volume which has the signature.

I/O time-out and I/O queue depth

Make sure to set the I/O time-out and I/O queue depth values for the DE volumes as specified in the applicable disk array configuration guide.

Partition size

Make sure to specify the correct partition size for the DE volumes as specified in the applicable disk array configuration guide. If the partition size for -A or -B volumes is smaller than the mainframe volume size, the open system host may not be able to access data to the end of the extent of these volumes.

For Solaris, use the following partition sizes for the DE volumes, and use 2 (two) for the number of alternate cylinders:

Partition sizes for Solaris,

LVI	Cylinder # for data cylinder extent
3390-3A	0 - 3345
3390-3B	0 - 3339
3390-3C	0 - 3345
3380-KA	0 - 2661
3380-KB	0 - 2655
3380-KC	0 - 2661
OPEN-3	0 - 3335
OPEN-8	0 - 9963
OPEN-K	0 - 2540
OPEN-9	0 - 10013
OPEN-E	0 - 19756
OPEN-L	0 - 19012
OPEN-M	0 - 7108

Partition sizes for Sun Solaris,

LVI	Cylinder # for data cylinder extent
3390-3A	0 - 3345
3390-3B	0 - 3339
3390-3C	0 - 3345
OPEN-3	0 - 3335
3390-9A	0 - 10035
3390-9B	0 - 10017
3390-9C	0 - 10035
3390-LA	0 - 32763
3390-LB	0 - 32760

Note: For further information on Solaris cylinder partition sizes, refer to the *HP StorageWorks Disk Array XP Operating System Configuration Guide for Sun Solaris.*

Volume labels

A DE volume with a volume label cannot be shared between open system platforms which use volume labels.

Sharing DE volumes between open system platforms

		IBM AIX	HP-UX	Sequent Dynix	Windows 2000 2003 NT	HP Tru64 UNIX	Sun Solaris	NCR SVR4	Linux
No label	IBM AIX	OK	OK	OK	OK	OK	OK	OK	OK
	HP-UX	OK	OK	OK	OK	OK	OK	OK	OK
	Sequent Dynix	OK	OK	OK	OK	OK	OK	OK	OK
	Linux	OK	OK	OK	OK	OK	OK	OK	OK
Label write option	HP Tru64 UNIX	OK	OK		OK	*	*		
	Windows2000/ 2003/NT	OK	OK	OK	OK	*	*	*	OK
Label auto-write	Sun Solaris	OK	OK	OK	OK	OK	OK		
	NCR SVR4	OK	OK	OK	OK	OK	NO	NO	NO

*Sharing allowed only if volume has no label.

HP-UX and IBM AIX

HP-UX and IBM AIX do not use volume labels, so DE volumes can always be shared with these platforms.

When installing an IBM AIX system, note:

- When the User ID is not the root, a patch is required.
- AIX V5.1can be used with 64-bit FAL.

Please contact your IBM technical representative for assistance.

Tru64 UNIX and Windows

Labels are optional for Tru64 UNIX and Windows 2000/2003/NT, so DE volumes can be shared with these platforms only if they have no label.

Sun Solaris

Sun Solaris always writes volume labels, so DE volumes can never be shared between these two platforms, but can be shared with the other platforms (HP, IBM, Tru64 UNIX, Windows) as long as they do not have labels.

Note: Sun Solaris may display the following warning messages when formatting and labeling an DE volume. This is normal, and the user can ignore these messages.

```
Warning: error writing VTOC
Warning: no backup labels
Write label failed
```

Access privileges

For UNIX hosts, make sure to set up the desired access privileges for each DE volume (e.g., using groups and/or chmod command). Please refer to the OS user documentation for information on access permission rights. For Windows 2000/2003/NT, Administrator access is required to access the DE volumes.

Mainframe host

On the S/390 host, make sure to initialize and write the VTOC for each MTO and OTM volume to enable the S/390 host to access the volumes. The ICKDSF media maintenance utility can be used to perform these tasks.

Installing the FAL/FCU software

This section explains how to install the Data Exchange FAL/FCU software on UNIX and NT systems. The UNIX instructions are in the next section. For NT instructions, see "Installing FAL/FCU on Windows NT/2000/2003" (page 36).

Also see "Entering the DE license key code" (page 37). The procedure is the same for both UNIX and NT systems.

Installing FAL/FCU on UNIX-based platforms

Note that there are separate instructions for 32-bit and 64-bit systems.

32-bit installation

To install the 32-bit FAL/FCU software on a UNIX-based platform:

1. If FAL/FCU version 01-01-36 or later is already installed (by **cpio** command), you do not need to deinstall it. The new installation will overwrite the previous version.

   If FAL/FCU version 01-01-24 or earlier is already installed (by setup program), you must deinstall this older version first. To deinstall:

   a) Log in with the same user ID that was used to install the old FAL/FCU software.

   b) If FAL/FCU was installed from FD using the setup program, see **Deinstallation** to remove FAL/FCU.

   c) If FAL/FCU was installed from DAT, or if you can't find your FAL/FCU installation FD, move to the directory **fcu/fal.o/dataset.h** and remove FAL/FCU by entering:

   ```
   # rm fcu fal.o dataset.h $HOME/FcuMf
   ```

   If you cannot find the directory, you can use the following procedure:

   ```
   # find / -name “fcu” -print
   # find / -name “fal.o” -print
   ```

```
# find / -name “dataset.h” -print
# find / -name “FcuMf” -print
```

2. Log in to the system as **root**.
3. Insert the FAL/FCU CD-ROM into the drive. Verify that the device file for the CD-ROM drive exists. ***Note:*** For Sun Solaris, do not use **volcheck** if the CD-ROM device file is not available for auto-mount.
4. Make sure the following six directories exist on the open system host. If not, create the directories using the **mkdir** command (e.g., **# mkdir /usr/lib/X11/app-defaults**).

   ```
   /usr
   /usr/lib
   /usr/bin
   /usr/lib/X11
   /usr/include
   /usr/lib/X11/app-defaults
   ```

5. Move to the **root** directory.
6. Copy the FAL/FCU software from the installation CD-ROM as follows:

   ```
   # cpio -iBmuv < CD_device_file_name/d
   ```

   Note: Use full device file name. Wildcards will not work.
7. For Sun Solaris you must set a path to the resource file for each FAL/FCU user as follows:

   a) For C shell, add the following line to the end of the **.cshrc** file in the home directory. If **.cshrc** does not exist, create it and enter the following line:

   ```
   setenv XFILESEARCHPATH
   /usr/lib/X11/app-defaults/%N:$XFILESEARCHPATH

   export XFILESEARCHPATH
   ```

   Note: Add these two lines to the file **.profile** in your home directory, when it is not in the common desktop environment. If **.profile** does not exist, create it.

   b) For non-C shell, add the following two lines to the end of the **.dtprofile** file in the home directory. If **.dtprofile** does not exist, create it and enter the following lines:

XFILESEARCHPATH=/usr/lib/X11/app-defaults/%N:$XFILESEARCHPATH
export XFILESEARCHPATH

c) You must log out and log back in to implement these changes.

8. For Tru64 UNIX you must uncompress the FAL/FCU program:

   # uncompress /usr/bin/fcu.Z

9. Remove the CD-ROM from the drive.
10. Log out, and then log in again.

Note: When the DE Code Converter is installed, the **libuoc.*** file is replaced with the Code Converter library (the extension varies according to OS). Before installing Code Converter, save libuoc.* with an alias.

64-bit installation

To install the 64-bit FAL/FCU software on a UNIX-based platform:

1. Log-in as "root".
2. Place the CD-ROM in the drive and mount it.
3. Check to see if the following directories currently exist. If they do not, create them as follows:

   All Platforms: **/usr**, **/usr/lib**

   Solaris: **/usr/lib/sparcv9**

   HP-UX: **/usr/lib/pa20_64**
4. Move to the root directory.
5. Copy 64-bit FAL from the CD-ROM.
6. A file or directory can be viewed using the correct file name given at mounting. To view a directory, use one of the following procedures according to platform:

   Install 64-bit FAL after confirming a directory name and a file name by using the **ls** command.

   - For HP-UX (11.0):

#cpio -iBmuv
<(MountPoint)/PROGRAM/FAL64/HP_UX/HP_UX.CPI

- For Solaris (Solaris 7,8, 9):

  #cpio -iBmuv
  (MountPoint)/PROGRAM/FAL64/SOLARIS/SOLARIS.CPI

- For AIX (AIX 4.3.3):

  #cpio -iBmuv
  <(MountPoint)/PROGRAM/FAL64/AIX/AIX4/AIX.CPI

- For Red Hat Linux AS3.0/IA64:

  #cpio -iBmuv
  <(MountPoint)/PROGRAM/FAL64/LINUX/LINUX.CPI

7. Remove the CD-ROM from the drive.
8. Log-out once and log-in again.

Deinstallation

To deinstall FAL/FCU:

1. Log in to the system as **root**.
2. Remove the FAL/FCU for 32-bit files using the **rm** command as follows, or string the commands:

```
# rm /usr/bin/fcu
# rm /usr/bin/fcunw
# rm /usr/include/dataset.h
# rm /usr/lib/libfal.
# rm /usr/lib/libuoc.
# rm /usr/lib/X11/app-defaults/FcuMf
# rm /usr/bin/mfformat
# rm /usr/bin/allocds
```

3. Remove the FAL/FCU for 64-bit files using the **rm** command as follows, or string the commands:

```
# rm /usr/bin/fcunw
# rm /usr/include/dataset.h
# rm /usr/lib/libfal64.*
```

# rm /usr/lib/libuoc64.*
# rm /usr/bin/mfformat64
# rm /usr/bin/allocds64

4. When deinstalling 32-bit FAL/FCU version 01-01-41 and later, remove the following file:

   # rm /usr/bin/listvol

5. When deinstalling 64-bit FAL/FCU version 01-01-41/00 and later, remove the following files:

   #rm /usr/bin/ppkeyset64
   #rm /usr/bin/autoppkeyset64

6. When deinstalling 32-bit FAL/FCU version 01-01-45 and later, remove the following files:

   # rm /usr/lib/libfal.ver
   #rm /usr/bin/ppkeyset
   #rm /usr/bin/autoppkeyset

After Installation

After FAL/FCU software installation, make sure to format each OTO volume using the FAL/FCU Formatter (FMT) utility on the UNIX host as explained in "Formatting OTO volumes using the FMT utility" (page 39). This enables the OTO intermediate datasets to be allocated.

Installing FAL/FCU on Windows NT/2000/2003

To install the FAL/FCU software on a Windows NT/2000/2003 platform:

1. If FAL/FCU is already installed, deinstall it before installing the new version.

   To deinstall FAL/FCU version 01-01-25 or later:

   a) Open the **Add/Remove Programs** control panel: click on **Start**, click on **Settings**, click on **Control Panel**, then double-click on **Add/Remove Programs**.

b) Select **FCU** in the list of installed programs, click on the **Add/Remove** button, and then follow the instructions on screen to complete the deinstallation.

FAL/FCU versions 01-01-24 and earlier do not support the Windows system deinstaller. To deinstall version 01-01-24 or earlier, delete the folder which contains the FAL/FCU software components (**fcu.exe**, **fal.obj**, and **dataset.h**).

2. Insert the FAL/FCU installation CD-ROM into the drive.
3. Run **setup.exe** and follow the instructions on screen.

Note: For Windows NT/2000/2003, if the **Installed Directory** has a directory name using a "space" character, enter the following:
<license key> fal.dll falmt.dll

After Installation

After FAL/FCU software installation, make sure to format each OTO volume using the FAL/FCU Formatter (FMT) utility on the Windows NT host as explained in "Formatting OTO volumes using the FMT utility" (page 39). This enables the OTO intermediate datasets to be allocated.

Entering the DE license key code

The license key for DE is entered by command on the server system after FAL/FCU has been installed. A license key is required for each server and for each different server type. Each key is associated with a specific disk array (defined by serial number).

For UNIX platforms, enter the command for the type of UNIX you are using from the command line:

- **HP-UX**
  - **32-bit**: ppkeyset <License key> /usr/bin/fcunw /usr/bin/fcu /usr/lib/libfal.sl
  - **64-bit**: ppkeyset64 <License key> /usr/bin/fcunw64 /usr/lib/pa20_64/libfal64.sl

- **Solaris**
  - **32-bit**: ppkeyset <License key> /usr/bin/fcunw /usr/bin/fcu /usr/lib/libfal.so.1
  - **64-bit**: ppkeyset64 <License key> /usr/bin/fcunw64 /usr/lib/sparcv9//libfal64.so.1
- **AIX**
  - **32-bit**: ppkeyset <License key> /usr/bin/fcunw /usr/bin/fcu /usr/lib/libfal.a
  - **64-bit**: ppkeyset64 <License key> /usr/bin/fcunw64 /usr/lib/libfal64.a
- **Digital UNIX/HP Tru64 UNIX/DYNIX/ptx**
  - 32-bit: ppkeyset <License key> /usr/bin/fcunw /usr/bin/fcu /usr/lib/libfal.so
- **Linux**
  - **32-bit**: ppkeyset <License key> /usr/bin/fcunw /usr/lib/libfal.so.1
  - **64-bit**: ppkeyset <License key> /usr/bin/fcunw64 /usr/lib/libfal64.so.1

Formatting OTO volumes using the FMT utility

After the FAL/FCU software has been installed on the open system host, you can format the OTO volumes using the FAL/FCU Formatter (FMT) utility. This enables you to allocate OTO intermediate datasets.

FMT for UNIX is a UNIX command executed from the command line.

FMT for Windows NT/2000/2003 is a GUI.

The FMT utility defines the size of the OPEN-*x* volume in cylinders. The maximum number of cylinders allowed by FMT are shown in the table below.

FMT utility values

Emulation Type	**01-XX-47 or earlier**	**01-XX-YY/ZZ not LUSE**	**01-XX-YY/ZZ LUSE**	**01-XX-YY/2x not LUSE**	**01-XX-YY/2x LUSE n=number of volumes (see *Note 2*)**
OPEN-3	3331	3331	5818	3331	(min 3338*n-7, 65534)
OPEN-8)	5818	5818	5818	9959	(min 9996*n-7, 65534)
OPEN-9	5818	5818	5818	10009	(min 10016*n-7, 65534)
OPEN-K) (XP128 only)	2536	2536	5818	2536	(min 2543*n-7, 65534)
OPEN-E	Not supported	5818	5818	19752	(min 19759*n-7, 65534)
OPEN-L	Not supported	5818	5818	49429	(min 49439*n-7, 65534)
OPEN-M (XP128 only)	Not supported	5818	5818	63976	(min 63999*n-7, 65534)
OPEN-V	Not supported	Not supported	Not supported	65534	(min(Vc*n*128/96-7,65534) (see Note 3)

Note 1: XX = 1 or 2; YY, x = number; ZZ<20

Note 2: For the Solaris system, the data cylinder must be less than or equal to 32767. When using a LUSE volume, the geometry parameter is different, so the number of cylinders should be calculated as follows:

Cylinder (specified to FAL formatting) <= (A*B*C) / (15*96) – 5

A: Head (Geometry parameter)

B: Block/Track (Geometry parameter)

C: cylinder (Geometry parameter)

Note 3: Vc = OPEN-V cylinder value (MAX.:49160 cylinders using FAL)

The FMT utility can be used on standard-size OPEN-*x* volumes and on Virtual LVI/LUN (VIR) volumes. ***Note:*** The VIR OPEN-*x* devices can also be called custom volume size (CVS) devices (e.g., OPEN-3-CVS). When formatting a VIR OPEN-*x* LU, use the number of cylinders defined for VIR minus seven (e.g., use 993 cylinders for a VIR device defined with 1000 cylinders). The cylinder size is: one cylinder = 15 tracks, one track = 96 sub-blocks, one sub-block = 512 bytes.

The table below shows the relationship between block length and available write capacity per track. The actual data capacity per cylinder = (write capacity per track) × (15 tracks).

Relation between block length and write available

Block Length by Allocater = (A) (Bytes)	**Write Available Data per Track (Bytes)**	**Block Length by Allocater = (A) (Bytes)**	**Write Available Data Per Track (Bytes)**	**Block Length by Allocater = (A) (Bytes)**	**Write Available Data Per Track (Bytes)**
23477 - 32760	(A) × 1	1589 - 1684	(A) × 22	565 - 596	(A) × 44
15477 - 23476	(A) × 2	1493 - 1588	(A) × 23	533 - 564	(A) × 45
11477 - 15476	(A) × 3	1397 - 1492	(A) × 24	501 - 532	(A) × 46
9077 - 11476	(A) × 4	1333 - 1396	(A) × 25	469 - 500	(A) × 47
7477 - 9076	(A) × 5	1269 - 1332	(A) × 26	437 - 468	(A) × 48
6357 - 7476	(A) × 6	1205 - 1268	(A) × 27	405 - 436	(A) × 49
5493 - 6356	(A) × 7	1141 - 1204	(A) × 28	373 - 404	(A) × 50
4821 - 5492	(A) × 8	1077 - 1140	(A) × 29	341 - 372	(A) × 51

Block Length by Allocater = (A) (Bytes)	Write Available Data per Track (Bytes)	Block Length by Allocater = (A) (Bytes)	Write Available Data Per Track (Bytes)	Block Length by Allocater = (A) (Bytes)	Write Available Data Per Track (Bytes)
4277 - 4820	(A) × 9	1045 - 1076	(A) × 30	309 -340	(A) × 52
3861 – 4276	(A) × 10	981 - 1044	(A) × 31	277 - 308	(A) × 53
3477 –3860	(A) × 11	949 - 980	(A) × 32	245 - 276	(A) × 54
3189 – 3476	(A) × 12	917 - 948	(A) × 33	213 - 244	(A) × 55
2933 – 3188	(A) × 13	853 - 916	(A) × 34	181 - 212	(A) × 56
2677 – 2932	(A) × 14	821 - 852	(A) × 35	149 - 180	(A) × 57
2485 – 2676	(A) × 15	789 - 820	(A) × 36	117 - 148	(A) × 58
2325 – 2484	(A) × 16	757 - 788	(A) × 37	85 - 116	(A) × 59
2165 – 2324	(A) × 17	725 - 756	(A) × 38	53 - 84	(A) × 60
2005 – 2164	(A) × 18	693 - 724	(A) × 39	21 - 52	(A) × 61
1877 – 2004	(A) × 19	661 - 692	(A) × 40	1 - 20	(A) × 62
1781 – 1876	(A) × 20	629 - 660	(A) × 41	—	—
1685 – 1780	(A) × 21	597 - 628	(A) × 42	—	—

Note: The write available data per track includes the four-byte RL information and four-byte BL information for each record. When transferring variable-length records, make sure to take this required extra space into account.

Formatting OPEN-*x* volumes in UNIX

Caution *The FMT utility erases all data on the OPEN-x LUN being formatted. If necessary, back up the data on the OPEN-x LUNs prior to using FMT utility for formatting.*

To format an OPEN-*x* volume using the FMT utility in UNIX-based systems:

1. Log in to the system as **root**.
2. Enter the following command at the UNIX command line prompt:

# mfformat -d *devname* -v *VOLSER* [-p *primary_cylinders*]

-d devname: Specify the raw device name (for example, "/dev/rdsk/c0t1d2" for HP-UX) of the OPEN-*x* volume being formatted. This parameter is required. Make sure to use the same raw device name for this volume in the OTO volume definition file.

-v VOLSER: Specify the VSN of the volume being formatted (A-Z, 0-9, @, #, \). Use only uppercase letters and do not use any spaces or symbols other than @, #, and \. This parameter is required. Make sure to use the same volser for this volume in the OTO volume definition file.

-p primary_cylinders: Specify the number of primary cylinders (from decimal 2 through 5818). This parameter is required for custom-size volumes but is optional for standard-size volumes. If this parameter is omitted, the default value of max cylinders is used. The maximum cylinders are:

OPEN-3 = 0-3331
OPEN-8 = 0-9959 ()
OPEN-9 = 0-10009
OPEN-K = 0-2536 ()
OPEN-E = 0-19752
OPEN-L = 0-49429
OPEN-M = 0-63976 ().

When LUSE is set, the default value is the maximum value in a single volume.

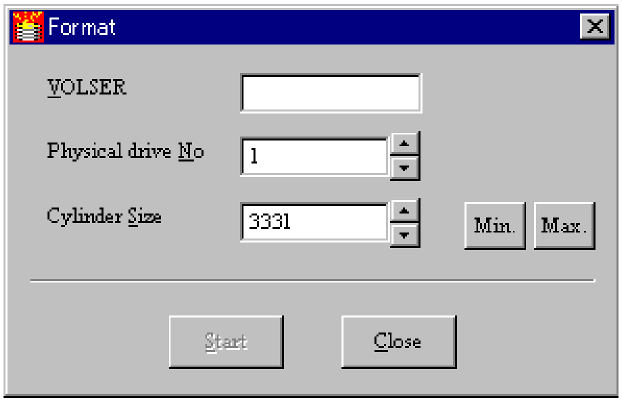
3. If the FMT format operation could not be started due to an error condition, the **Format check error** message is displayed. If the FMT format operation did not complete successfully, an error message is displayed. Remove the error condition, and retry the operation. See "Troubleshooting" (page 167) for more information on error conditions.

Formatting OPEN-*x* volumes in Windows

Caution *The FMT utility erases all data on the OPEN-x LUN being formatted. If necessary, back up the data on the OPEN-x LUNs prior to using FMT utility for formatting.*

To format an OPEN-*x* volume using the FAL/FCU FMT utility for Windows NT/2000/2003:

1. Log in to the system as administrator.
2. Double-click on the **Format** icon to open the Format panel.



Physical drive No: 1
Cylinder Size: 3331

3. Enter the six-character volume serial number for the OPEN-*x* volume being formatted in the **VOLSER** field. Make sure to use the same volser for this volume in the OTO volume definition file.
4. Specify the physical drive number (device number) for the OPEN-*x* volume being formatted in the **Physical drive No** field. Make sure to use the same physical drive number for this volume in the OTO volume definition file.
5. Specify the number of cylinders for the OPEN-*x* volume in the **Cylinder Size** field. The **Min.** button enters 2 cylinders and the **Max.** button enters 5818 cylinders.

   If the OPEN-*x* volume is standard size (e.g., OPEN-3), use the maximum size of 5818 cylinders.

   If the OPEN-*x* volume is custom size (e.g., OPEN-3*n VIR device), use the following value: **(# of cylinders defined for VIR) – 7**. For example, if the VIR OPEN-*x* volume is defined with 1000 cylinders, enter 993 in the **Cylinder Size** field.

   Note: The maximum size for the Allocater is 4369 cylinders.
6. Verify that the entries are correct and click on **Start**.

7. When the **Format confirmation** appears click on **OK** to perform the requested format operation or select **Cancel** to cancel your request.
8. When the format operation completes successfully, the **Format complete** message is displayed.

   Possible errors

   If the specified volser has already been used, an error message is displayed.

   If the format operation could not be started due to an error condition, the **Format check error** message is displayed.

   When **Format check error** occurs, the formatting operation has not started and the original condition has been kept. When another message is indicated, the formatting process has already started. The data on the volume has already been initialized. Remove the error condition and format the volume again.

   If the format operation did not complete successfully, one of the following error messages is displayed:

Open error! (n)	Open process error on specified volume.
Seek error! (n)	Seek process error on specified volume.
Read error! (n)	Read process error on specified volume.
Write error! (n)	Write process error on specified volume.
Close error! (n)	Close process error on specified volume.

   If you execute over the maximum number of cylinders on Windows 2003, the message **Format failed … Format check error** appears after about a minute. It should be clear when formatting has completed.
9. When you are finished formatting OPEN-*x* volumes for use as OTO volumes, select **Close** to close the **Format panel** and exit the FMT utility.

Host access to DE volumes

The user must manage access to the MTO and OTM volumes to prevent illegal I/O access contention between the S/390 and open system hosts. These DE volumes cannot be accessed concurrently by the S/390 and open system hosts and must be varied offline from the S/390 host during DE operations. The DE volumes should not contain any regularly accessed data and should be dedicated to data exchange operations to avoid accidental overwriting of data.

Note: Please note the following restrictions for certain operating systems:

- **UNIX**: OTM can run several different datasets simultaneously.
- **Windows NT/2000/2003**: OTM can run several different datasets simultaneously.
- **AIX**: Since volumes are reserved during accessing, OTM cannot run several different datasets simultaneously.

VSE support and transfer conditions

Data Exchange supports the VSE mainframe operating system as follows.

OS version		**Data Exchange Version**
VSE 2.3 and lower		Before
VSE 2.5 and upper		After

MTO and OTM transfer conditions for VSE

This section defines transfer conditions when performing MTO or OTM transfers with a mainframe using the VSE operating system.

VSE 2.3

When you use MTO and OTM for the dataset allocated by VSE 2.3, you must specify record format (RF), record length (RL), and block length (BL) using the VSE parameters provided in the table below.

VSE 2.5

When you use MTO and OTM for the dataset allocated by VSE 2.5, Data Exchange can transfer data without specifying VSE parameters.

Most transfers are allowed. However, note that special conditions apply in some circumstances.

		Record Format			
OS and Data Exchange Version	**VSE Parameter Specified**	**Fixed non-block length**	**Fixed block length**	**Variable non-block length**	**Variable block length**
VSE 2.3 and previous	Yes	Yes	Yes	Yes	Yes
	No	No	No	No	No
VSE 2.5 and later (Data Exchange version 01-03-59 and later)	Yes	Yes	Yes	Yes	Yes
	No	Yes	Yes	Yes	Yes

As this table illustrates, all parameters must be specified when using VSE 2.3 and previous versions, or data cannot be transferred. When using VSE 2.5 or later, data can be transferred whether or not parameters are specific, with restrictions as listed in Notes 1-4 below.

Note 1:

It is possible to transfer data between correct dataset attributes (5 ≤RL ≤ BL-4). The data transfer is valid only if the VSE parameters are as shown:

RL ≤ 32756

BL ≤ 32760

BL = RL+4

For the following dataset attributes, the data transfer is invalid if user does **not** specify the VSE parameter value as shown above.

RL ≤ 32756

BL ≤ 32760

For the following dataset attributes, the data transfer is invalid if user does **not** specify the VSE parameter value between RL and BL values shown in #1 and #2.

RL ≤ 32756

BL ≤ 32760

RL = BL

s#1: RL (Input value for VSE parameter) = RL (value on VTOC) + 4 ≤ 32756

#2: BL (Input value for VSE parameter) = BL (value on VTOC) + 8 ≤ 32760

Note 2:

It is possible to transfer data between the correct dataset attributes (5 ≤ RL ≤ BL-4). The data transfer is invalid if the VSE parameter is **not** the following value:

RL (Input value for VSE parameter) = RL (value on VTOC) + 4 ≤ 32756

BL (Input value for VSE parameter) = BL (value on VTOC) + 8 ≤ 32760

Note 3:

It is possible to transfer data between the correct dataset attributes ($BL \leq RL+4 \leq 32760$). The data transfer is invalid if the RL and BL values on VTOC do **not** match to the following condition.

$BL \leq RL+4 \leq 32760$

When the RL and BL values on VTOC are such that $RL > 32756$ and $BL > 32760$, Data Exchange manages the data as RL=32756 and BL=32760.

When the RL and BL values on VTOC are such that $RL \leq 32756$, $BL \leq 32760$ and $RL = BL$, Data Exchange manages the data as shown below:

RL(Data Exchange internal value) = RL(value on VTOC) + 4 $\leq$ 32756

BL(Data Exchange internal value) = BL(value on VTOC) + 8 $\leq$ 32760

Note 4:

It is possible to transfer data between the correct dataset attributes ($BL \leq RL+4 \leq 32760$). The data transfer is invalid if RL and BL values on VTOC do **not** match to the following condition:

$BL \leq RL+4 \leq 32760$

When the RL and BL values on VTOC are such that $RL \leq BL \leq 32752$, Data Exchange manages the data as shown below:

RL(Data Exchange internal value) = RL(value on VTOC) + 4

BL(Data Exchange internal value) = BL(value on VTOC) + 8

If you create a user program by using the dataset attribute exchange rule, you need to use the following functions:

datasetGetFileInformation()

datasetGetFileInformationEx()

Compiler requirements

The compiler requirements for each operating system are shown in the table below. If the development environment differs from that specified here, your results may differ.

Operating System/Platform	Compiler

RedHat Linux 7.2
(Kernel version 2.4.7-10)
and
RedHat Linux AS2.1 and AS3.0
	gcc(Ver.2.96-98)
	glibc(Ver.2.2.4-13)

Solaris	Work Shop Compilers 4.2.1
Windows NT 4.0 service Pack 6a	Visual C++ 6.0 SP4
HP-UX 10.20	HP 92453-01 A.10.32.03 HP C Compiler
AIX 4.3	C for AIX 4.4
AIX5L	C for AIX, Version 5.0
Tru64	Digital UNIX Compiler Driver 3.11 DEC C V5.6-075 on Digital UNIX V4.0 (Rev. 878)

64-bit version FCU

Data Exchange has a 64-bit version FCU. Its use is the same as for the 32-bit version FCU. ***Note***: The 64-bit version FCU does not have a GUI.

3

Overview of FCU operations

This section explains how the File Conversion Utility works, including details about MTO, OTM, and OTO operations.

FCU file transfer options

The File Conversion Utility (FCU) requires identification of the transfer direction (MTO or OTM), the source files, and the target files. (An OTO operation consists of one OTM operation followed by one MTO operation.)

FCU provides the following options for file transfer operations:

- Code conversion (CC)
- Padding (PAD)
- Delimiters (DEL)
- Record description word (RDW)
- VSE record (VSE)
- Empty file (EMP)

Note: Multiple Volume Dataset is supported only for MTO and only on mainframes that use the MVS operating system.

Code conversion (CC) option

The code conversion option can be used for both MTO and OTM operations. You can use either the default EBCDIC-ASCII code conversion table included with FCU or your own code conversion table.

Always use code conversion when transferring text files between mainframe and open systems. Do not use code conversion when transferring binary data files. Code conversion is available (**EcA** option) but not recommended for OTO file transfers.

Default code conversion table

When the default table is specified, FCU performs EBCDIC-to-ASCII code conversion for MTO operations and ASCII-to-EBCDIC code conversion for OTM operations.

The default EBCDIC-ASCII code conversion table is the ACM standard table (not CACM). Appendix B, "EBCDIC-ASCII code conversion" (page 201) provides the code conversion information for the default table shown below. If the default code conversion table does not yield the desired results, you can create your own code conversion table. Please refer to the IBM code tables for detailed information about EBCDIC-ASCII code conversion.

Default EBCDIC-ASCII Code Conversion Table for FCU

H L	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL (00)	DLE (10)	DS (80)	(90)	SP (20)	& (26)	- (2D)	(BA)	(C3)	(CA)	(D1)	(D8)	{ (7B)	} (7D)	\ (5C)	0 (30)
1	SOH (01)	DC1 (11)	SOS (81)	(91)	(A0)	(A9)	/ (2F)	(BB)	a (61)	j (6A)	(E5)	(D9)	A (41)	J (4A)	(9F)	1 (31)
2	STX (02)	DC2 (12)	FS (82)	SYN (16)	(A1)	(AA)	(B2)	(BC)	b (62)	k (6B)	s (73)	(DA)	B (42)	K (4B)	S (53)	2 (32)
3	ETX (03)	DC3 (13)	(83)	(93)	(A2)	(AB)	(B3)	(BD)	c (63)	l (6C)	t (74)	(DB)	C (43)	L (4C)	T (54)	3 (33)
4	PF (9C)	TM (9D)	BYP (84)	PN (94)	(A3)	(AC)	(B4)	(BE)	d (64)	m (6D)	u (75)	(DC)	D (44)	M (4D)	U (55)	4 (34)
5	HT (09)	(85)	LF (0A)	RS (95)	(A4)	(AD)	(B5)	(BF)	e (65)	n (6E)	v (76)	(DD)	E (45)	N (4E)	V (56)	5 (35)
6	LC (86)	BS (08)	ETB (17)	UC (96)	(A5)	(AE)	(B6)	(C0)	f (66)	o (6F)	w (77)	(DE)	F (46)	O (4F)	W (57)	6 (36)
7	DEL (7F)	IL (87)	ESC (1B)	EOT (04)	(A6)	(AF)	(B7)	(C1)	g (67)	p (70)	x (78)	(DF)	G (47)	P (50)	X (58)	7 (37)
8	GE (97)	CAN (18)	(88)	(98)	(A7)	(B0)	(B8)	(C2)	h (68)	q (71)	y (79)	(E0)	H (48)	Q (51)	Y (59)	8 (38)

Default EBCDIC-ASCII Code Conversion Table for FCU																
9	RLF (8D)	EM (19)	(89)	(99)	(A8)	(B1)	(B9)	‘ (60)	i (69)	r (72)	z (7A)	(E1)	I (49)	R (52)	Z (5A)	9 (39)
A	SMM (8E)	CC (92)	SW (8A)	(9A)	(D5)	! (21)	(CB)	: (3A)	(C4)	^ (5E)	(D2)	(E2)	(E8)	(EE)	(F4)	(FA)
B	VT (0B)	CUI (8F)	CUI (8B)	CU3 (9B)	. (2E)	$ (24)	, (2C)	# (23)	(C5)	(CC)	(D3)	(E3)	(E9)	(EF)	(F5)	(FB)
C	FF (0C)	IFS (1C)	(8C)	DC4 (14)	< (3C)	* (2A)	% (25)	@ (40)	(C6)	(CD)	(D4)	(E4)	(EA)	(F0)	(F6)	(FC)
D	CR (0D)	IGS (1D)	ENQ (05)	NAK (15)	((28)	) (29)	_ (5F)	’ (27)	(C7)	(CE)	[(5B)	] (5D)	(EB)	(F1)	(F7)	(FD)
E	SO (0E)	IRS (1E)	ACK (06)	(9E)	+ (2B)	; (3B)	> (3E)	= (3D)	(C8)	(CF)	(D6)	(E6)	(EC)	(F2)	(F8)	(FE)
F	SI (0F)	IUS (1F)	BEL (07)	SUB (1A)	| (7C)	~ (7E)	? (3F)	“ (22)	(C9)	(D0)	(D7)	(E7)	(ED)	(F3)	(F9)	(FF)

	Bit Positions	
	Hi	Lo
ASCII	8765	4321
EBCDIC (IBM)	0123	4567

User-defined code conversion table

The user-defined code conversion table must be a binary data file created by placing the target code values in the offset positions that correspond to the source code values.

Item	Requirement
Size	256 bytes
Format	Binary data
Code length	One byte (two-byte codes cannot be converted)
File name	The following sequences of characters cannot be used in the file name: EA EcA EkJ No If the file name for the code conversion table contains any of these sequences, FCU will ignore the file and use the default table instead.

Pipe function

This function transfers data entries from the mainframe to the application program or the utility program for UNIX systems using a “named pipe”. When this function is used, a mainframe dataset can be transferred to an open system. This is a much faster way to transfer data than the code conversion method.

Using the pipe function in UNIX systems

A “named pipe” is a special file that is used to transfer data between unrelated processes. One or more processes write to it, while another process reads from it. Named pipes are visible in the file system and may be viewed with ‘**ls**’ like any other file. (Named pipes are also called “fifes; this term stands for `First In, First Out'.) Named pipes may be used to pass data between unrelated processes, while normal (unnamed) pipes can only connect parent/child processes (with some exceptions). Named pipes are strictly unidirectional, even on systems where anonymous pipes are bidirectional (full-duplex).

Pipe function details

FCU can carry out data transmission to a pipe file. A user application opens and reads this pipe file and a direct data transmission is attained between the application and FCU. There is no need for it to be output as a file on an HDD.

A named-pipe name is specified in a parameter definition file as the output file name. **“Pipe=yes”** needs to be specified as an option. Under these conditions, FCU will open a pipe file with the specified output file name, and will transmit data to it.

If the specified file exists as a standard UNIX file, FCU re-creates a pipe file using the same name and the UNIX file is deleted. Since FCU only inputs in data to a pipe, the FCU function needs to obtain the data via a user application. If data remains in the pipe, FCU will stop and processing does not progress to the next step. The FCU function has an built-in timer. If the application does not continue receiving data, FCU will send an error message after a certain set time and it will progress to the next logical process.

Pipe function timeout value

FCU waits for a “Read Data Entries” status message. A time-out error will be reported if the TIME OUT VALUE is not set appropriately. The TIME OUT VALUE should be set in the WAIT_TIME_VALUE environment variable. The limits are 0~1440 seconds (0 = unlimited). The default value is 10 when the timeout value is undefined.

The following examples illustrate the use of the WAIT_TIME_VALUE environment variable. ***Note:*** After setting the variable, log out and log in again to establish the variable’s value.

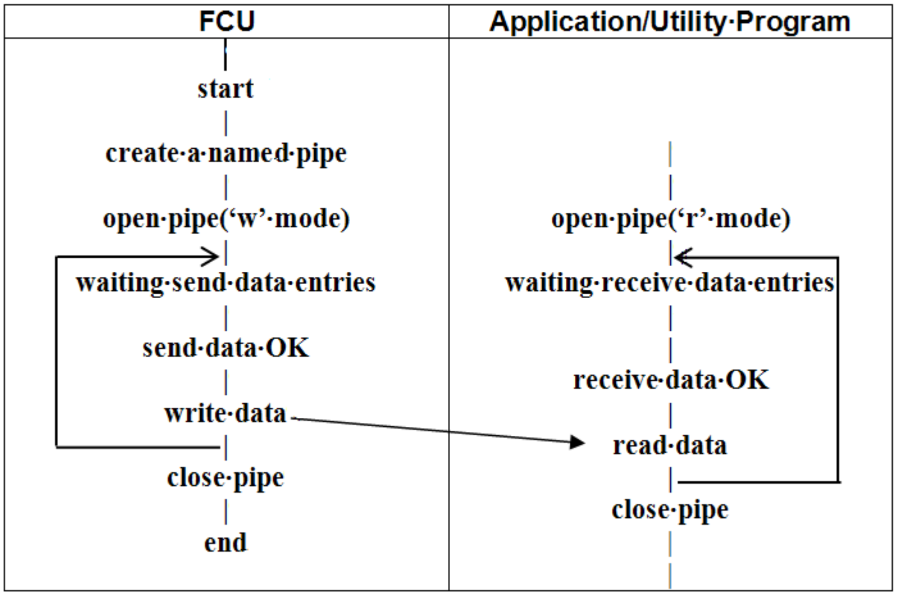
Example 1: For C shell:

- Add “setenv WAIT_TIME_VALUE 300” to the file“.cshrc” in the home directory.
- If “.cshrc” does not exist, create it and add the “setenv” line.

Example 2: For non-C shell:

- Add “WAIT_TIME_VALUE=300”
- Add “export WAIT_TIME_VALUE”
- These two commands must be added to the file “.dtprofile” in the home directory. If “.dtprofile” does not exist, create it and add the lines.

The figure below illustrates the pipe function process.



Padding (PAD) option

The padding option can be used for MTO with variable-length source datasets and for OTM with fixed-length target datasets. When the padding option is specified for MTO, FCU adds padding to each source data record, so that the length of the each record equals the maximum record length. When padding is specified for OTM, FCU adds padding to each source data entity, so that the length of the each target record equals the record length defined for the target dataset. FCU transfers the data entities with padding to the target file/dataset. FCU cannot extract padding from files or datasets.

The type of padding added by FCU depends on whether code conversion was also requested:

- **Padding with code conversion (text files)**. When padding and code conversion are both specified, FCU adds **spaces** to the short data entities as needed.
- **Padding without code conversion (binary data files)**. When padding is specified but code conversion is not, FCU adds **0x00** to the short data entities as needed.

Note: If you use MTO with padding, the data cannot be transferred back to the original S/390 dataset (the OTM target dataset will not be compatible with the original dataset). If you use OTM with padding, the delimiter option is required.

Delimiter (DEL) Option

The delimiter option can be used for both MTO and OTM operations and enables variable-length records to be transferred between platforms without losing compatibility with the original dataset. When the delimiter option is specified for MTO, FCU adds the specified delimiter to the end of each data entity in the source file, and then extracts and transfers the data entity with delimiter to the open system target file. When the delimiter option is specified for OTM, FCU extracts each data entity preceding the specified delimiter and transfers the data entities without delimiters to the target dataset.

The type and length of the delimiter added (or recognized and extracted) by FCU depends on the open system platform:

- For UNIX-based platforms, you must specify either a carriage return (CR) or a line feed (LF). The length of this delimiter is one byte.
- For Windows 2000/2003/NT, you must specify a CR + LF. The length of this delimiter is two bytes.

Note: Do not use the delimiter option for OTM if the source file contains the same character as the delimiter but used for a purpose other than delimiting data entities. If you do, FCU will interpret the specified delimiter character as delimiters, which can create a target dataset with corrupt records or generate an error condition.

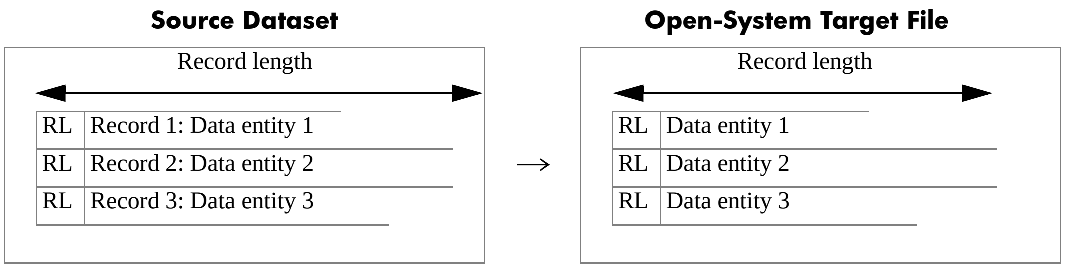
Note: If you use MTO with delimiter (no padding) for variable-length records, you will be able to transfer the data back to the original S/390 dataset later using OTM.

Empty file (EMP) option

The empty file option can be used for both MTO and OTM operations. When the empty file option is specified, FCU processes an empty source file instead of returning an error. An empty S/390 dataset is a dataset which has no records or only EOF records. An empty open system file is a file which has a file size of 0 bytes. When an empty S/390 dataset is processed, the open system target file size = 0. When an empty open system file is processed, the target dataset will contain only EOF records.

Record description word (RDW) option

The record description word option can only be used for MTO operations on variable-length source datasets. The example below shows an MTO operation with the RDW option specified. When the RDW option is specified, FCU adds the record description word in binary code to the head of each record in the source dataset, and then transfers the data entity with record length bytes to the open system target file. The CC, PAD, and DEL parameters must be **No**; if not, FCU returns an error. If the RDW option is specified for a fixed-length source dataset, FCU ignores the RDW option.



MTO with the RDW Option

Note: If you use MTO with RDW, the data cannot be transferred back to the original S/390 dataset (the OTM target dataset will not be compatible with the original dataset)

VSE record (VSE) option

The VSE record option must be used for MTO and OTM operations involving VSE datasets. The VTOC of a VSE dataset does not specify the record format (RF), record length (RL), or block length (BL) of the dataset. The VSE record option enables the user to specify these values so that FCU can process source/target VSE datasets. If you do not specify the VSE record option for a VSE dataset, FCU will return an error. If you specify the VSE record option and the RF, RL, and BL are also specified in the VTOC, FCU will process the dataset if the RF, RL, and BL values are the same, or return an error if the RF, RL, and BL values are not the same. The VSE record option does not apply to ALC-generated intermediate datasets.

MTO operations

An MTO operation transfers the data from an S/390 dataset on an DE volume to an open system file on an open system LUN. The object data entities are those contained in all records between the beginning of the file and the end of the file. The end of a dataset is the EOF record or the end of the final extent. The end of an open system file is the EOF. The MTO source file must be located on an DE -B or -A volume on the disk array. If the specified MTO target file does not exist, FCU automatically creates the target file during the MTO operation. If the specified MTO target file already exists, FCU requests confirmation to overwrite the target file (unless the **-nc** option is specified).

The FCU software performs the MTO data transfer operations. FCU supports both fixed-length and variable-length record formats and provides the following options for MTO data transfer: code conversion, padding, delimiter, empty file, record description word, and VSE record. The types of MTO operations are:

- MTO with fixed-length record format
- MTO with variable-length record format.

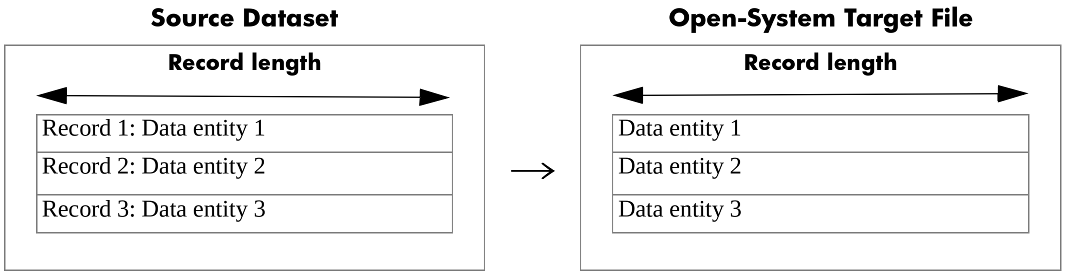
The table below specifies the record format requirements for each type of MTO operation. A fixed-length source dataset can only be transferred to a fixed-length target file, with or without delimiters. Padding cannot be added to a fixed-length source file. A variable-length source dataset can be transferred to a variable-length or fixed-length target file, depending on the padding option, and delimiters can also be added if desired.

			Record Format Requirements		
FCU Direction	**Padding**	**Delimiters**	**Source Dataset**	**Target File**	**See Example**
MTO	N/A	No	Fixed-length	Fixed-length	Table (page 62)
MTO	N/A	Yes	Fixed-length	Fixed-length	Table (page 62)
MTO	No	No	Variable-length	Variable-length	Table (page 63)
MTO	Yes	No	Variable-length	Fixed-length	Table (page 64)
MTO	No	Yes	Variable-length	Variable-length	Table (page 64)
MTO	Yes	Yes	Variable-length	Fixed-length	Table (page 65)

MTO with fixed-length record format

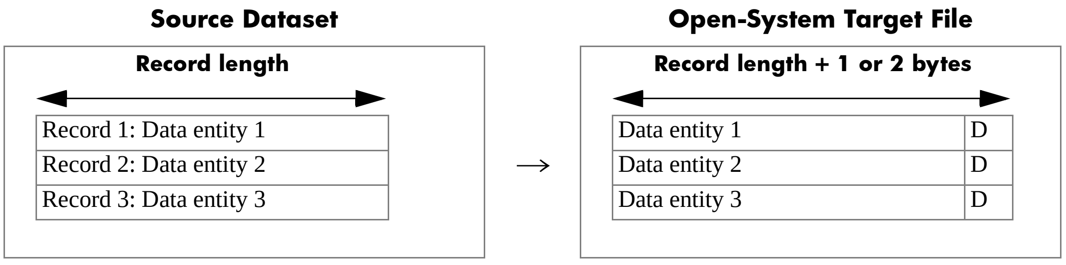
Each fixed-length record in an S/390 dataset includes only the fixed-length data entity. The record length defined for a fixed-length dataset equals the actual length of each data entity. The padding option cannot be used for MTO with fixed-length records.

No padding, no delimiters. The illustration below shows an MTO operation for a fixed-length source dataset. FCU extracts and transfers the data entities to the open system target file. The length of each data entity in the target file equals the record length defined for the source dataset.



MTO with fixed-length records: no padding, no delimiters

With delimiters. The illustration below shows an MTO operation with delimiters (D) for a fixed-length source dataset. FCU extracts and transfers the data entities to the open system target file and adds the requested delimiter to the end of each data entity. The resulting length of each data entity in a equals the original data record plus one byte for the delimiter for UNIX and two bytes for the delimiter for Windows 2000/2003/NT target files.



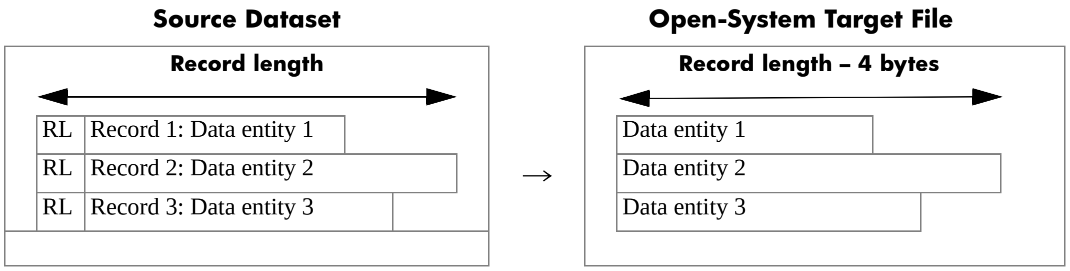
MTO with fixed-length records: delimiters

MTO with variable-length record format

Each variable-length record in an S/390 dataset includes a four-byte RL field and the variable-length data entity. The record length defined for a variable-length dataset equals the maximum allowable record length. ***Note:*** If you want to be able to transfer the data back to the original S/390 dataset later, you must use MTO without padding and with delimiters.

No padding, no delimiters. The illustration below shows an MTO operation without padding or delimiters for a variable-length source dataset. FCU extracts and transfers only the data entities to the target file. The RL fields are not transferred. The resulting length of each data entity in the target file is equal to or less than the maximum record length minus four bytes (for the RL field).

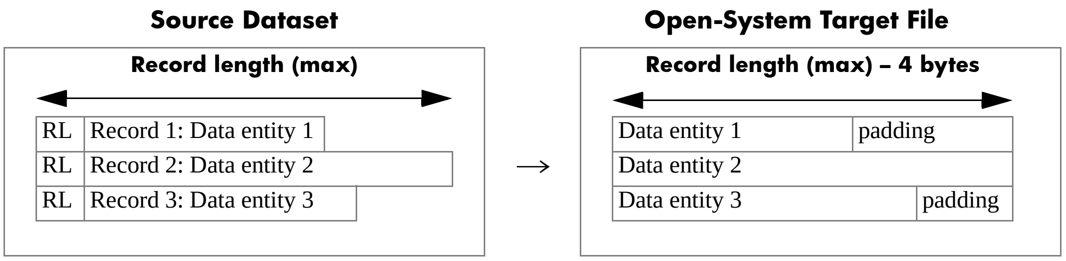
Note: If you plan to transfer the data back to the original dataset later using OTM, use MTO with delimiters.



MTO with variable-length records: no padding, no delimiters

With padding. The illustration below shows an MTO operation with padding. MTO with padding requires a variable-length source file and produces a fixed-length target file. FCU adds padding to the source records as needed so that the length of each record equals the maximum record length. FCU then extracts and transfers the data entities with padding to the open system target file. The RL fields are not transferred. The resulting length of each data entity in the target file equals the maximum record length minus four bytes (for the RL field).

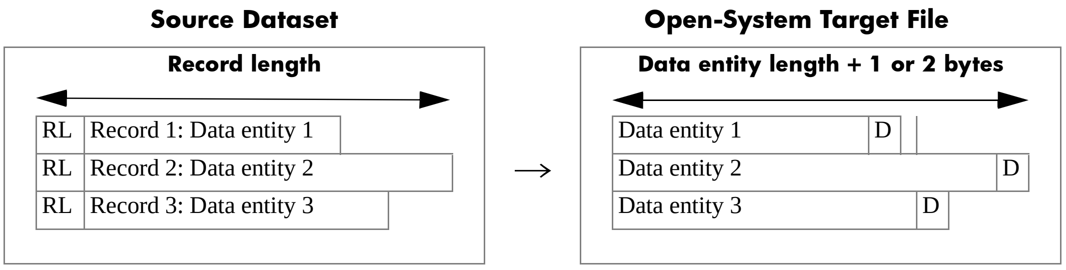
Note: If you use MTO with padding, you will not be able to transfer the data back to the original dataset later using OTM.



MTO with variable-length records: padding

With delimiters. The illustration below shows an MTO operation with delimiters (D) for a variable-length source dataset. FCU extracts and transfers the data entities to the open system target file and adds the requested delimiter to the end of each data entity. The RL fields are not transferred. The resulting length of each data entity in a UNIX target file equals the original data entity length plus one byte for the delimiter. The resulting length of each data entity in a Windows 2000/2003/NT target file equals the original data entity length plus two bytes for the delimiter.

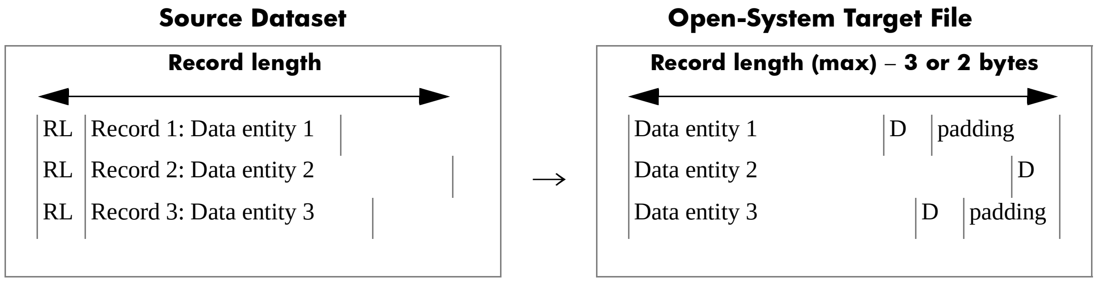
Note: If use MTO with delimiters and without padding, you will be able to transfer the variable-length records back to the original dataset later using OTM.



MTO with variable-length records: delimiters

With padding and delimiters. The illustration below shows an MTO operation with padding and delimiters (D). MTO with padding and delimiters requires a variable-length source file and produces a fixed-length target file. FCU adds the appropriate delimiter to each data entity, adds the appropriate amount of ‘padding’ so that each record equals the maximum record length, and then extracts and transfers the data entities with padding and delimiters to the open system target file. The RL fields are not transferred.

Note: If you use MTO with padding and delimiters, you will not to be able to transfer the records back to the original dataset later (the padding cannot be removed).


Source Dataset
Record length
RL Record 1: Data entity 1
RL Record 2: Data entity 2
RL Record 3: Data entity 3
Open-System Target File
Record length (max) – 3 or 2 bytes
Data entity 1 D padding
Data entity 2 D
Data entity 3 D padding

MTO with variable-length records: padding and delimiters

The resulting length of each data entity in a UNIX target file equals the maximum record length minus three bytes (minus four for the RL, plus one for the delimiter). The resulting length of each data entity in a Windows 2000/2003/Window NT target file equals the maximum record length minus two bytes (minus four for the RL, plus two for the delimiter).

MTO with multiple volume datasets

Multiple Volume Dataset is supported only for MTO, when the version is 01-XX-50/ YY(XX=01 or 02 or 03) or higher.

Note: Multiple Volume Dataset is not supported for OTM. A multiple volume definition file (multidef.dat) is necessary in current directory. FAL will check Dataset Serial number, Data set serial number and Last volume containing data in this data set in Data set indicators on VTOC DSCB1.

Note: FAL/FCU supports two kinds of mainframes, MVS and VSE. For Multiple Volume Dataset, FAL/FCU is only supported on MVS.

	Data Set Serial Number			
	1	**1**	**Except 1**	**Except 1**
	Last volume containing data in this data set in Data set indicators.			
VTOC DSCB1/ Action	**On**	**Off**	**On**	**Off**
OTM	OK(OK)	OK(OK)	NG(OK)	NG(OK)
MTO for single volume(*1)	OK(OK)	NG(OK)	NG(OK)	NG(OK)

AIX shared open function

To share volumes on multiple AIX operating systems:

- For versions 01-xx-59 and earlier, when one AIX OS opens a volume, the other AIX OS cannot open the volume. This is because the AIX OS reserves the volume when it opens the it.
- For 01-xx-60 and later, it is possible to share an FileExchange volume across multiple AIX operating systems by specifying the environment variable:

  ```
  FAL_NO_RESERVE.
  ```

The following table shows the relationship between shared volumes and versions.

Object	Version	Description
FAL for AIX (32/64 bit)	01-xx-59 and earlier	Cannot open a shared volume from multiple AIX OSs.
	01-xx-60 and later	Can open a shared volume from multiple AIX OSs by specifying the environment variable.

To define environment variables:

- By specifying the environment variable (FAL_NO_RESERVE), you can select **shared open** or **exclusive open**. The table below shows the relationship between the environment variable (FAL_NO_RESERVE) and open mode.

FAL_NO_RESERVE	Open Mode
No definition of environment variable	Exclusive Open (Original mode)
ON	Shared Open
OFF	Exclusive Open
Other	Exclusive Open

AIX reserve function

AIX reserved retry function

For versions 01-XX-61 and earlier, an AIX system error will occur when you use FX to send data to a reserved volume on another system.

For versions 01-XX-63 and later, you can retry sending data to the reserved volume by specifying the wait time and retry count for the environment variable of the other system. can send data again when the reserved volume is released by the other system. The following table shows the AIX reserve functions.

Environment Variable or Function	Description
FAL_RETRY	To set retry function, specify the environment variable.
FAL_RETRY_COUNT	To set retry count, specify the environment variable. (FX attempts top send data while in reserve status.)
FAL_RETRY_WAIT_TIME	To set the interval, specify the environment variable. (FX attempts to send data while in reserved status.)
FAL_RETRY_TARGET	To set system error codes, specify the environment variable. (System error codes triggers.)
Retry function	The retry function in invoked when AIX commands trigger a system error (open, close, read, write, seek, flush).
Out put retry log function	FX outputs a retry log (FAL_Error log) when it executes retry.

Defining the environment variable

- **Retry Function:** You can set the retry function by specifying the environment variable (FAL_RETRY).
  - If there is no definition of environment variable, the Retry function is **disabled**.

  - If the environment variable is defined as **ON**, the Retry function is **enabled**.
  - If the environment variable is defined as **OFF**, the Retry function is **disabled**.
  - If the environment variable is defined as **other**, the Retry function is **disabled**.
- **Retry Counts Function:** You can set the number of retry function instances by specifying the environment variable (FAL_RETRY_COUNT). attempts to send data during reserved status.
  - The default count number is 10, where no definition of the environment variable is given.
  - The range of the FAL_RETRY_COUNT variable is 1-600.
  - If other values outside the range are given, the default value applies.
- **Retry Wait Time Function:** You can set the wait time interval by specifying the environment variable (FAL_RETRY_WAIT_TIME). attempts to send data during reserved status.
  - The default is 1 second, where no definition of the environment variable is given.
  - The range of the FAL_RETRY_WAIT_TIME variable is 1-60 seconds.
  - If other values outside the range are given, the default value applies.
- **Retry target function:** You can set system error codes, which are retry triggers, by specifying the environment variable (FAL_RETRY_TARGET).
  - You can set up to 5 system error codes separated by commas. If you set six or more system error codes, will ignore all after the fifth.
  - If no environment variable definition is given, an error code 16(EBUSY) will appear.
  - You can set up to 5 system error codes, separated by commas.

- If other values outside the range are given, the 16(EBUSY) error code applies.

Output retry-log function

FX outputs a retry-log to the FAL_Error log file when it executes **retry**. The following shows the retry-log format:

```
Mon Nov  8 16:21:23 2004 : root : err=16 open Retry(1) at 12345 : 01-03-58/21 PID=1234
VSN:DSN
```

The entry log format includes the following:

- data
- user name
- system error code
- functions (open or close or read or write or seek or flush)
- retry counts
- number of source code line
- version
- process ID
- target data-set

System errors

Error code 16 EBUSY indicates that system resources are busy. The reserved volume will be opened and error code 16 will appear.

OTM operations

An OTM operation transfers the data from an open system file on an DE volume to a target dataset on an S/390 volume. The OTM source file must be located on an DE -C or -A volume on the disk array. FCU does not automatically create the OTM target dataset. The target dataset must be created and properly formatted prior to beginning the OTM operation.

The FCU software performs the OTM data transfer operations. FCU version 01-01-40 or later is required for VSE target datasets. FCU supports fixed-length and variable-length record formats for OTM operations. FCU provides the following options for OTM operations:

- code conversion
- padding (01-01-41 and later)
- delimiter
- empty file
- VSE record.

The record description word option cannot be used with OTM. FCU automatically extracts delimiters from OTM source files, but cannot add delimiters to OTM source files. FCU can add padding only to variable-length OTM source files. FCU cannot extract padding from OTM source files.

The types of OTM operations are:

- OTM with fixed-length record format (page 72) and
- OTM with variable-length record format (page 75).

The table below specifies the record format requirements for each type of OTM operation.

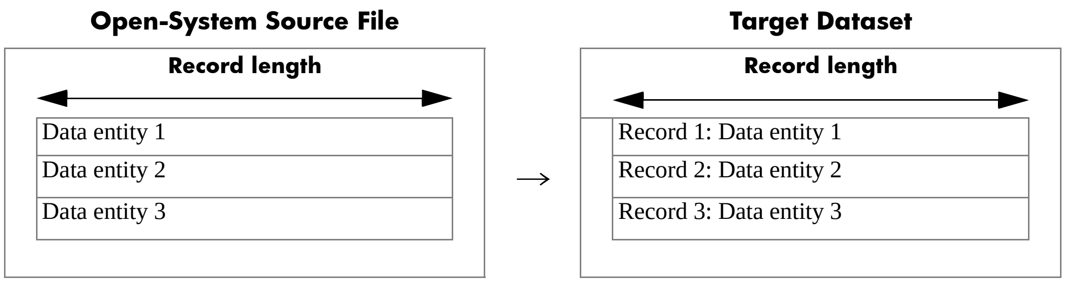
	Record Format Requirements			
FCU Directio n	Source File		Target Dataset	See Figure
OTM	Fixed-length: no padding, no delimiters	$\longrightarrow$	Fixed-length	Table (page 73)
OTM	Fixed-length containing padding	$\longrightarrow$	Fixed-length	Table (page 73)
OTM	Fixed-length containing delimiters	$\longrightarrow$	Fixed-length	Table (page 74)
OTM	Fixed-length containing padding and delimiters	$\longrightarrow$	Fixed-length	Table (page 74)
OTM	Variable-length: with delimiters	$\longrightarrow$	Variable-length	Table (page 75)
OTM	Variable-length: with padding and delimiters	$\longrightarrow$	Fixed-length	Table (page 76)

An open system source file with fixed-length data entities can only be transferred to a fixed-length target dataset. An open system source file with variable-length data entities must have delimiters and can be transferred to a variable-length or fixed-length target dataset. If the source file contains padding from a previous MTO transfer operation, the padding is transferred to the target dataset along with the data. If the source file contains delimiters, the delimiters are not transferred to the target dataset.

Note: Do not update the volume that is transferred directly by the OTM.

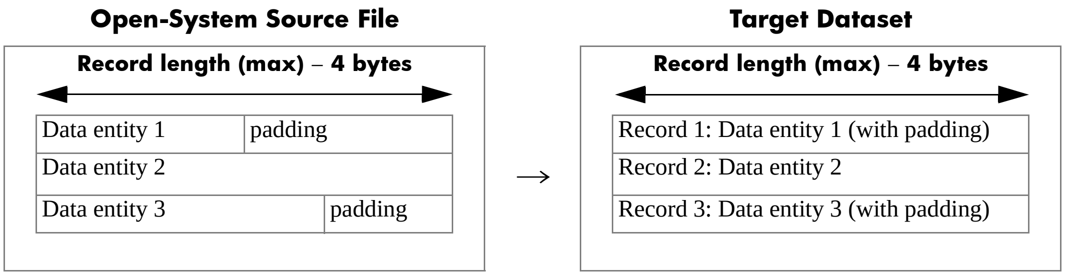
OTM with fixed-length record format

No padding, no delimiters. The illustration below shows an OTM operation for a fixed-length source file without padding or delimiters. The target dataset must have fixed-length record format with record length set to the actual length of each data entity. If the data entity length does not exactly match the record length defined for the target dataset, FCU aborts the operation and reports an error.



OTM with fixed-length records: no padding, no delimiters

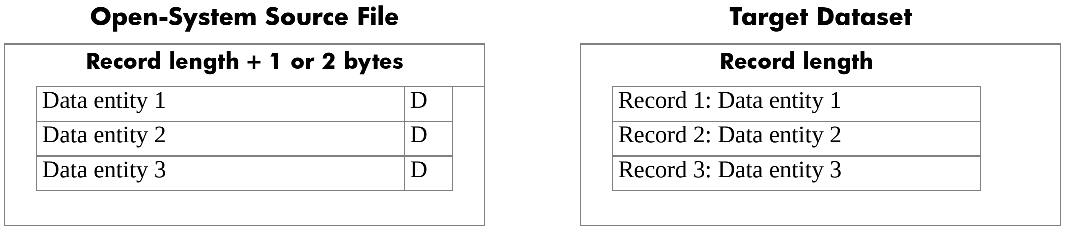
With padding. The illustration below shows an OTM operation for a fixed-length source file with padding from a previous MTO transfer. The original MTO dataset cannot be used as the OTM target dataset. FCU transfers the data entities including padding to the target dataset. The length of each data entity in the source file equals the maximum record length minus four bytes (for the RL field). The target dataset must have fixed-length record format with record length set to the maximum record length minus four bytes. If the length of any record (data entity plus padding) in the source file does not exactly match the record length defined for the target dataset, FCU aborts the operation and reports an error.


Open-System Source File
Record length (max) – 4 bytes
Data entity 1
padding
Data entity 2
Data entity 3
padding
Target Dataset
Record length (max) – 4 bytes
Record 1: Data entity 1 (with padding)
Record 2: Data entity 2
Record 3: Data entity 3 (with padding)

OTM with fixed-length records: padding

Note: FCU does not extract padding from OTM source files.

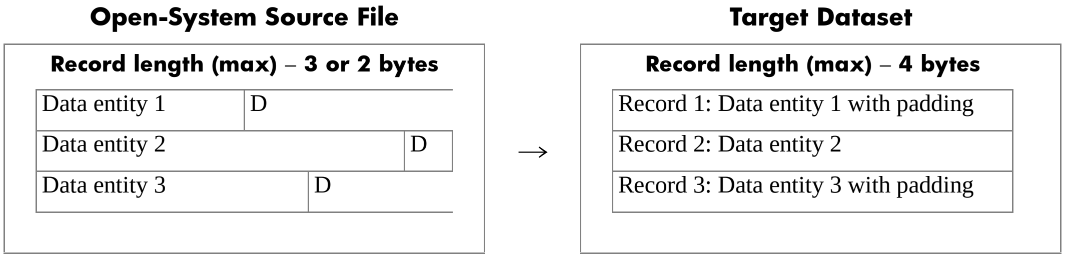
With delimiters. The illustration below shows an OTM operation for a fixed-length source file with delimiters from a previous MTO transfer. FCU extracts the data entities from the source file by record length and transfers them to the target dataset. The delimiters are not transferred. The target dataset must have fixed-length record format with record length set to the actual length of each data entity (without delimiter). If the length of any source data entity does not exactly match the record length defined for the target dataset, FCU aborts the operation and reports an error. If the delimiter is not found right after the data entity, FCU aborts the operation reports an error.



OTM with fixed-length records: Delimiters

Note: FCU does not add delimiters to OTM source files. If the OTM source file contains delimiters but you specify **No** for the delimiter option, the delimiters will be regarded as part of the data entities and will be transferred to the target dataset.

With padding and delimiters. The illustration below shows an OTM operation for a fixed-length source file with padding and delimiters from a previous MTO transfer. FCU removes the delimiters but not the padding and transfers the data entities with padding to the target dataset. The original variable-length dataset cannot be used as the target dataset for this transfer. The target dataset must have fixed-length record format with record length set to the maximum record length minus four bytes. If the length of any source data entity does not match the record length defined for the target dataset, FCU aborts the operation and reports an error.


Open-System Source File
Record length (max) – 3 or 2 bytes
Data entity 1
D
Data entity 2
D
Data entity 3
D
Target Dataset
Record length (max) – 4 bytes
Record 1: Data entity 1 with padding
Record 2: Data entity 2
Record 3: Data entity 3 with padding

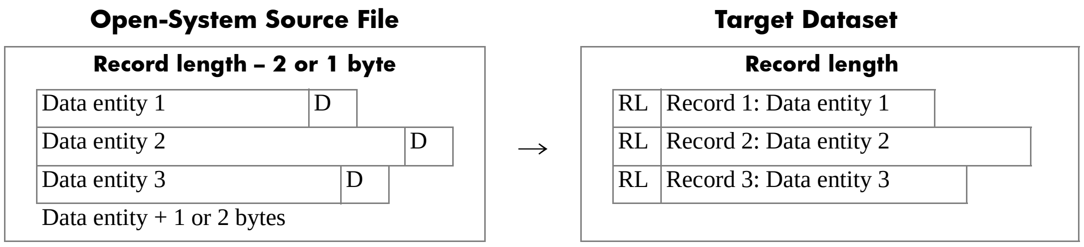
OTM with Fixed-Length Records: Padding and Delimiters

Note: FCU does not extract padding from OTM source files. If the OTM source file contains delimiters but you specify **No** for the delimiter option, the delimiters will be regarded as part of the data entities and will be transferred to the target dataset.

OTM with variable-length record format

OTM operations can be performed on variable-length source files only if delimiters have already been added to the source file (for example, from a previous MTO operation). If a variable-length source file without delimiters is processed, FCU will use the maximum record length to construct the target data entities, thereby corrupting the data and rendering the dataset unusable. FCU extracts but does not add delimiters to OTM source files.

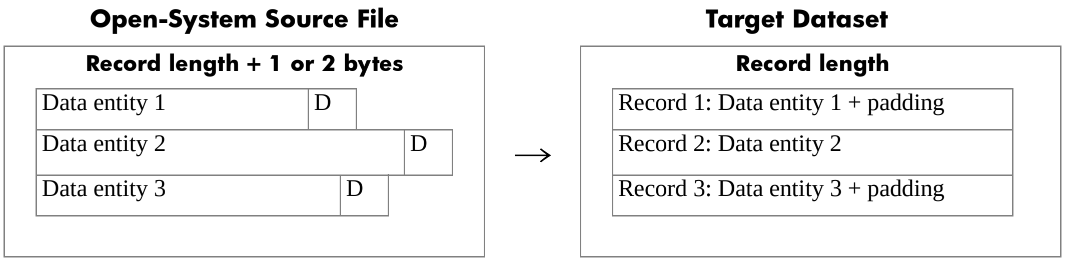
With delimiters. The illustration below shows an OTM operation for a variable-length source file with delimiters. FCU extracts and transfers the data entities to the target dataset, and automatically adds the four-byte RL field. The delimiters are not transferred. The target dataset must have variable-length record format.



OTM with Variable-Length Records: Delimiters

If the length of any data entity in a UNIX source file is greater than the maximum record length minus one byte (CR or LF delimiter), FCU aborts the operation and reports an error. If the length of any data entity in a Windows 2000/2003 or Windows NT source file is greater than the maximum record length minus two bytes (CR+LF delimiter), FCU aborts the operation and reports an error.

With padding and delimiters. The illustration below shows an OTM operation with padding for a variable-length source file with delimiters. FCU adds padding, extracts and transfers the data entities with padding to the target dataset, and automatically adds the four-byte RL field. The delimiters are not transferred. The target dataset must have fixed-length record format with record length defined as needed.



OTM with Variable-Length Records: Padding and Delimiters

If the length of any data entity in a UNIX source file is greater than the specified record length plus one byte (CR or LF delimiter), FCU aborts the operation and reports an error. If the length of any data entity in a Windows 2000/2003/NT source file is greater than the specified record length plus two bytes (CR+LF delimiter), FCU aborts the operation and reports an error.

OTO operations

OTO operations transfer data from source files on one open system platform to target files on another open system platform. Each OTO file transfer consists of two separate DE operations: first an OTM operation transfers the data in the source file to an intermediate dataset, and then an MTO operation transfers the data from the intermediate dataset to the target file. For users with the all-open systems (no attached S/390 host), the intermediate datasets are allocated on OPEN-*x* FMT volumes. The FMT utility enables you to format OPEN-*x* LUNs (standard or custom size) as OTO volumes. The ALC utility enables you to allocate intermediate datasets on the OPEN-*x* FMT volumes. For users with the multi platform systems, the intermediate datasets can be allocated on OPEN-*x* FMT volumes or on DE -A volumes, as desired. When you perform OTO operations which access OPEN-*x* FMT volumes, the OTO volume definition file must be available for use by FCU.

The FCU file transfer options (code conversion, padding, delimiters, etc.) can be used on the OTM and MTO sub-operations as needed.

- Code conversion is not available for OTO transfers.
- Padding can be used but will render the target file incompatible with the source file due to the change in record format from variable-length to fixed-length. If you use padding for the OTM operation, the target file can be transferred back to the same intermediate dataset but not back to the same source file. If you use padding for the MTO operation, the target file cannot be transferred back to the same intermediate dataset or back to the same source file.
- Delimiters can be used to enable bidirectional data transfers. When using delimiters, watch out for files which contain the same character as the delimiter (CR and/or LF) but used for purposes other than delimiting data entities. If you specify the delimiter option for OTM, FCU will interpret all occurrences of the specified delimiter character as delimiters, which can create a dataset with corrupt records or generate an error condition.
- The empty file option can be used to enable empty files to be processed. For example, if a source file specified in your OTO FCU

parameter definition file becomes empty, you can add the empty file option to the OTM/MTO operations on that file to enable FCU to process the FCU parameter definition file without errors.

- The RDW option is not normally used for OTO operations. If you use the RDW option (MTO operation only), you will not be able to transfer the data back to the same intermediate dataset.
- The VSE record option does not apply to OTO operations which access ALC-generated intermediate datasets on OPEN-*x* FMT volumes. The only time you would use the VSE option is when transferring a file between open system platforms via a VSE dataset on a -A DE volume. In this case, you must use the VSE record option for both transfers (OTM/MTO).

I/O access contention

The DE volumes can only be accessed by open system hosts using the FAL/FCU software. The S/390 hosts have normal read/write access to the -B and -A volumes, read-only access to the -C volumes, and no access at all to the OPEN-*x* FMT volumes. The open system hosts have read/write access to the -C, -A, and OPEN-*x* FMT volumes and read-only access to the -B volumes. The open system hosts must use FAL/FCU to access all DE volumes.

Caution *Concurrent access to the DE volumes by the S/390 and open system hosts is not supported. The user is responsible for managing access to DE volumes to avoid I/O contention between the S/390 and open system hosts. Since FCU accesses only the VTOC area of the DE -B volumes, catalog or security control functions cannot be used to provide access control for the 3390-3B/9B/LB and 3380-KB volumes (3390-9B/LB is only; 3380-KB is only).*

The S/390 host can issue a **reserve** command to reserve a volume for exclusive use. The S/390 **reserve** command prevents access by all other hosts, including all other S/390 hosts and all open system hosts. The open system host can also reserve a volume to exclude I/Os issued by other systems. The open system **reserve** command prevents access by all other open system hosts, but S/390 hosts still have normal access to MTO and OTM volumes reserved by open system hosts. These **reserve** commands affect DE operations as follows:

- **Reserved by S/390 host**. When an DE volume is reserved by the S/390 host, DE operations cannot be performed on that volume, because the FAL/FCU access from the open system host will terminate unsuccessfully. Open system access other than read or write I/Os can be executed successfully.

  Note: Open system access to an S/390-reserved volume may complete successfully if the open system retries the operation after the reserve is released. However, since the time interval before a retry varies depending on the open system platform and the S/390 application that issued the reserve, the success of retry operations on reserved volumes cannot be guaranteed.

- **Reserved by open system host**. When an DE volume is reserved by the open system host, DE operations can be performed only from the host that reserved the volume. DE operations from any other open system host will terminate unsuccessfully. Open system reserve does not affect S/390 access to the DE volume.
- **Unreserved**. When an DE volume is not reserved by any S/390 or open system host, DE operations can be performed from any open system host using FAL/FCU. All S/390 hosts and all open system hosts have access to unreserved volumes.

  The user should implement exclusive access control and job coordination at the system level for the DE volumes. The user should also take the following steps to avoid I/O contention problems for the DE volumes:
- **Open system access**. When the open system host needs to access an DE volume, vary the volume and its channel path offline from all S/390 hosts.
- **S/390 access**. When the S/390 host needs to access an DE volume, stop all open system access to the corresponding LU. For AIX, vary off the volume group. For Windows 2000/2003/NT, use **unaccess**. Do not use any open system program which accesses unmounted LUNs (e.g., AIX SMIT, HP-UX SAM, NT Disk Administrator).

Bidirectional data transfer

DE supports bidirectional data transfer for both fixed-length and variable-length S/390 datasets. Bidirectional data transfer involves transferring data from S/390 datasets to open system files and then back to the original S/390 datasets again. The requirements for bidirectional data transfer are:

- For all MTO operations, do not specify the record description word (RDW) option. If the RDW option is specified for an MTO data transfer, the subsequent OTM target dataset will not be compatible with the original dataset.
- For MTO with fixed-length datasets, do not specify the delimiter option, since the data entities are extracted by length. If you add delimiters for the MTO transfer, the subsequent OTM target dataset will not be compatible with the original dataset.
- For MTO with variable-length datasets, you must add delimiters but not padding. If delimiters are not added or if padding is added for the MTO transfer, the subsequent OTM target dataset will not be compatible with the original dataset.
- For OTM operations, do not specify the delimiter option if the source file contains the same character as the delimiter (CR and/or LF) but used for purposes other than delimiting data entities. If you specify the delimiter option for OTM, FCU will interpret all occurrences of the specified delimiter character as delimiters, which can create a dataset with corrupt records or generate an error condition.

4

Preparing for FCU and FAL operations

Before you prepare for FCU and FAL operations on the open system host, make sure that your DE environment has been set up and defined correctly, including DE volume configuration, FAL/FCU software installation, and OTO volume formatting. After your DE environment is set up, you need to perform the following tasks to prepare for FCU and FAL operations:

- Create the DE volume definition files (page 84).
- Verify S/390 dataset requirements for MTO and OTM (page 88).
- Allocate the intermediate datasets for OTO (page 90). This is not required for FAL operations.
- Create one or more FCU parameter definition files (page 94). This is not required for FAL operations.

Creating the DE volume definition file

The DE volume definition file contains the volume association parameters for the DE volumes on the disk array. This file must be created before you can use FCU or FAL to access data on these volumes. The volume association parameters define the DE volume by associating the volume serial number (VSN or volser) with the open system device file for the same logical volume. The table below describes the DE volume association parameters. Other tables in this section show the structure and contents of the DE volume definition file for each supported platform.

Note: The same VSN can be defined in the volume definition file using VSN identification, and both volumes can be used by OTM and MTO. The definitions should contain 35 digits, using alpha (A-Z, @, #, and \) or numeric (0-9) characters.

The -A, -B, and -C DE volumes and the OPEN-*x*-OTO volumes can be defined in the same DE volume definition file. For example:

```
XXX/XXXXXX MVS01 3390-3A
YYY/YYYYYY VSN01 OPEN-3
end
```

DE Volume Association Parameters			
Number	**Name**	**Function**	**Enter**
1	Device file name	Specifies raw device (partition) name defined for open system.	Character-type device file name (e.g., **c1t0d2** for HP-UX, **c1t0d2s1** for Sun Solaris
2	VOLSER	Specifies logical volume defined for S/390.	Six-character volser (e.g., DE45). A volser can use the following characters: A-Z, 0-9, @, #, \
3	LVI or LU type (emulation)	Specifies LVI or LU type of DE volume.	Correct LVI/LU for DE volume: 3390-3A, -3B, -3C, 3380-KA, -KB, -KC (XP128 only), or OPEN-*x*-OTO. Make sure to define all OPEN-*x* FMT volumes in a separate file.
4	Carriage return	Marks end of parameter set.	Make sure to press the **Return** key (**Enter** key for Windows 2000/2003/NT) at the end of each line.
5	End of file	Marks end of parameter file.	end

To create the DE volume definition file:

1. Open a new empty text file.

   For UNIX-based systems, use the UNIX vi editor (e.g., **vi datasetmount.dat**).

   For Windows 2000/2003/NT systems, use any text editor, and make sure to use plain text.

   The file name must be **datasetmount.dat** (all lowercase), and the file must be located in the current working directory when you start FCU.

   If you are creating two DE volume definition files, use **datasetmount1.dat** and **datasetmount2.dat**, and remove the “1” or “2” from the desired file before starting FCU.

2. Add the volume association parameters for the DE volumes to the file. See the examples below for information that applies to your OS.

   a) Put at least one space between each parameter, and press the **Return** key at the end of each line to separate the parameter sets. All three parameters (device name, volser, LVI type) are case-sensitive. If you add comments to the file, make sure that each comment line starts with #. Make sure to enter **end** on the last line of the file.

   b) When you are done adding the volume association parameters for each DE volume to the volume definition file, save your changes and exit the text editor.

   DE Volume Definition File for Sun Solaris (MTO/OTM shown)

```
/dev/rdsk/cx1tx2dx3sx4   AAAAAA 3390-3A   MFN  MVS1
/dev/rdsk/cy1ty2dy3sy4   AAAAAA 3390-3A   MFN  VOS3
/dev/rdsk/cz1tz2dz3sz4   cccccc 3380-KB
/dev/rdsk/cw1tw2dw3sw4   dddddd 3380-KA
end
```

Note: x = controller number, y = SCSI target ID (TID), z = LUN, w = partition (or slice)

DE Volume Definition File for HP-UX (OTO shown)

```
/dev/rdsk/cx1tx2dx3   AAAAAA  3390-3A   MFN  MVS
/dev/rdsk/cy1ty2dy3   AAAAAA  3390-3A   MFN  VOS3
/dev/rdsk/cz1tz2dz3   cccccc  3380-KB
/dev/rdsk/cw1tw2dw3   dddddd 3380-KA
end
```

Note: In cxtydz, x = controller number, y = SCSI TID, z = LUN. In OPEN-x, x = 3, 8, K, E, L, M, 9 or V.

DE Volume Definition File for IBM AIX (MTO/OTM shown)

```
/dev/rhdiskn1   AAAAAA 3390-3A   MFN  MVS
/dev/rhdiskn2   AAAAAA 3390-3A   MFN  VOS3
/dev/rhdiskn3   cccccc 3380-KB
/dev/rhdiskn4   dddddd 3380-KA
end
```

Note: n = disk ID number (note that the first, second, and third drives are 0, 1, 2).

DE Volume Definition File for DIGITAL Tru64 /UNIX OTO shown)

```
/dev/rrzX1Y1Z1   AAAAAA  3390-3A  MFN   MVS
/dev/rrzX3Y3Z3   cccccc  3380-KB  MFN   VOS3
/dev/rrzX4Y4Z4   dddddd  3380-KA
end
```

Note: X = b through h = LUN1 through LUN7 (no letter is used for LUN0); Y = fibre bus number × 8 + SCSI TID; Z = partition = a through h. For example, rrzc18a = SCSI TID 2, LUN2 (partition a) on fibre bus 2.

Note: OPEN-x = 3, 8, K, E, L, M, 9 or V

DE Volume Definition File for Windows NT /2000/2003 systems (MTO/OTM shown)

```
\\.\PHYSICALDRIVE0  AAAAAA 3390-3A   MFN MVS
\\.\PHYSICALDRIVE1  AAAAAA 3390-3A   MFN VOS3
\\.\PHYSICALDRIVE2  cccccc 3380-KB
\\.\PHYSICALDRIVE3  dddddd 3380-KA
end
```

Note: n = disk ID number.

DE Volume Definition File for NCR UNIX

```
/dev/rdsk/cx1tx2dx3s0  AAAAAA    3390-3B  MFN  MVS
/dev/rdsk/cy1ty2dy3s0  AAAAAA    3390-3A  MFN  VOS3
/dev/rdsk/cz1tz2dz3s0  cccccc    3380-KB
/dev/rdsk/cw1tw2dw3s0  dddddd    3380-KA
end
```

DE Volume Definition File for DYNIX/ptx

```
/dev/rdsk/sdx1  AAAAAA  3390-3A  MFN  MVS
/dev/rdsk/sdx2  AAAAAA  3390-3A  MFN  VOS3
/dev/rdsk/sdx3  cccccc  3380-KB
/dev/rdsk/sdx4  dddddd  3380-KA
end
```

DE Volume Definition File for Linux

```
/dev/rsda   AAAAAA   3390-3A   MFN   MVS1
/dev/asdb   AAAAAA   3390-3A   MFN   VOS3
/dev/rsd    ccccccc  3380-KB
/dev/rsd    ddddddd  3380-KA
end
```

Verifying S/390 dataset requirements

FAL and FCU have specific requirements for the DE source and target datasets. The table below specifies the requirements for DE datasets. The FCU GUI allows the user to display the dataset attributes and verify the dataset requirements.

FCU for UNIX () also provides the **listvol** function to display S/390 dataset attributes without using the GUI, which is explained in "Using FCU without the GUI" (page 195). The OTM target dataset (which can also be an OTO intermediate dataset) must be created and properly configured before the DE operation is performed. FCU does not support automatic expansion of the extent during OTM operations. The DE ALC utility allocates intermediate datasets in accordance with the requirements specified below.

Item	**Requirement**
Dataset organization (DO) type	SAM (sequential-access method). DE does not support any other DO types (e.g., DAM, VSAM, PAM). If a non-SAM dataset is specified, DE will return an error. Multiple-volume datasets are not supported. DE can only process the portion within one logical volume.
Dataset name	No spaces. If DE encounters a space, it will accept the characters before the space as the dataset name and continue processing.
Record format (RF)	Fixed-length or variable-length record format. DE does not support undefined-length or spanned record formats. If an illegal RF is detected, DE will return an error. No key. If a record with a key is accessed, DE will return an error. For OTM, the record format of the target dataset must be preconfigured to match the record format of the data entities in the source file. For VSE source and target datasets, the VSE record option must be used to specify the RF.

Item	Requirement
Block length (BL)	Any length within the extent supported by the OS. If an illegal BL is detected, DE will return an error. For OTM, the block length of the target dataset must be preconfigured to match the block length of the data entities in the source file. For VSE source and target datasets, the VSE record option must be used to specify the BL.
Record length (RL)	Any length within the extent supported by the OS. If an illegal RL is detected, DE will return an error. ***Note:*** DE cannot process a variable-length dataset which includes a record with no data entity (RL = 4). For OTM, the record length of the target dataset must be preconfigured to match the record length of the data entities in the source file. For VSE source and target datasets, the FCU VSE record option must be used to specify the RL.
Track format	Standard record 0 (R0). DE cannot process tracks with nonstandard R0.
VTOC	For MVS: standard or index VTOC. For an index VTOC, DE ignores the index and accesses the entire VTOC sequentially. For VSE: the user must specify the RF, BL, and RL using the FCU VSE record option. ***Note:*** The FAL functions cannot be used on VSE datasets.
Database file	Direct access is not supported; must be converted to a SAM file.

Allocating OTO intermediate datasets using ALC

When you perform OTO operations using OPEN-*x* FMT volumes, you must allocate the intermediate datasets before starting the file transfer operations. The DE Allocater (ALC) utility can only be used on OPEN-*x* volumes which have already been formatted using the DE FMT utility (see "Overview of FCU operations" (page 51).

Note: The ALC utility for UNIX is a UNIX command executed from the UNIX command line. The ALC utility for Windows 2000/2003/NT systems is a GUI. The ALC utility for UNIX can only be used on volumes formatted with the FMT utility for UNIX. The ALC utility for Windows 2000/2003/NT systems can only be used on volumes formatted with the FMT utility for Windows 2000/2003/NT systems.

Note: The capacity of the intermediate dataset varies depending on block length. Use the information in the table in the section "Verifying S/390 dataset requirements" (page 88) to calculate the required size for the intermediate dataset. When transferring variable-length records, make sure to take the four-byte RL information and four-byte BL information for each record into account.

Unix systems

To allocate an OTO intermediate dataset using the ALC utility:

1. Log in to the system as **root**.
2. Enter the following command at the UNIX command line prompt:

   # allocds -d *devname* [-n *datasetname*] [-f *recform*] [-r *reclen*] [-b *blocklen*] [-c *cylinders*]

   Note: Enter only one value for each parameter. You can only allocate one dataset at a time.

   -d devname: Specify the raw device name of the OPEN-*x* volume on which the dataset is being allocated. This parameter is required and must be specified.

-n datasetname: Specify the name of the dataset being allocated (maximum forty-four characters: A-Z, 0-9, @, #, ., \). Use uppercase letters only, and do not use any spaces or symbols other than @, #, ., and \. This parameter is required. If not specified, ALC will return the residual capacity (free space) on the specified volume in number of cylinders.

-r recform: Specify the record format of the dataset being allocated: **F** (fixed-length and de-blocking), **FB** (fixed and blocking), **V** (variable and de-blocking), or **VB** (variable and blocking). This parameter is required. If not specified, the default value of **F** is used.

-r reclen: Specify the record length (decimal) of the dataset being allocated: **1 to 32760**. This parameter is required. If not specified, the default value of 4096 is used.

-b blocklen: Specify the block length (decimal) of the dataset being allocated.

When record format = F, block length = record length.

When record format = FB, block length = record length × N (N = integer).

When record format = V/VB, block length = record length + 4 or more.

This parameter is required. If not specified, the following default values are used:

When record format = F/FB, default block length = record length.

When record format = V/VB, default block length = record length + 4.

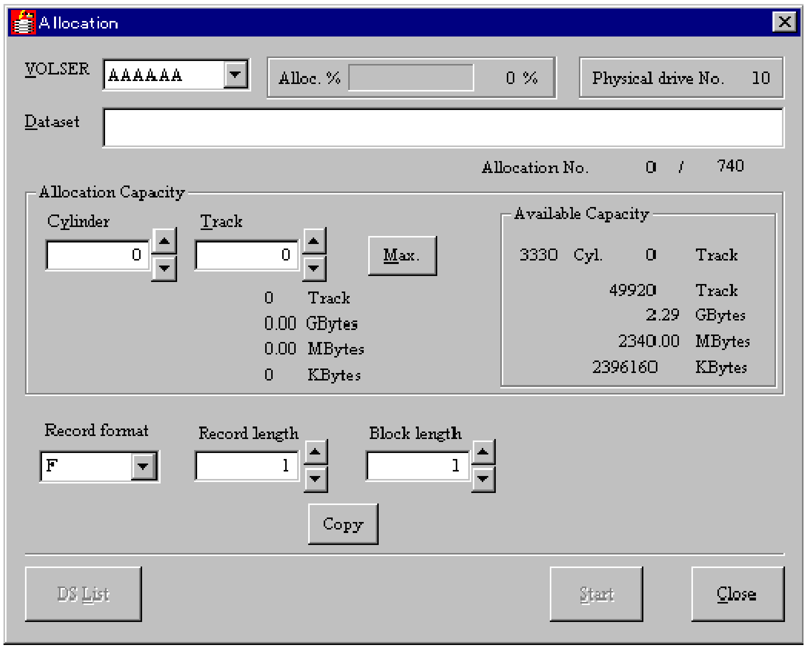
-c cylinders: Specify the size of the dataset being allocated in number of cylinders (decimal). This parameter is required. If not specified, the default value of 100 is used.

3. If the ALC allocate operation could not be started due to an error condition, the **Allocate check error** message is displayed. If the ALC allocate operation did not complete successfully, an error message is displayed. Remove the error condition, and retry the operation.

Windows systems

To allocate an intermediate OTO dataset using the ALC utility on Windows NT/2000/2003 systems:

1. Log in to the system as administrator.
2. Double-click on the **Allocate** icon to start the ALC utility and open the Allocation panel.



VOLSER AAAAAA
Alloc. % 0 %
Physical drive No. 10
Dataset
Allocation No. 0 / 740

Allocation Capacity
Cylinder 0
Track 0
0 Track
0.00 GBytes
0.00 MBytes
0 KBytes

Available Capacity
3330 Cyl. 0 Track
49920 Track
2.29 GBytes
2340.00 MBytes
2396160 KBytes

Record format F
Record length 1
Block length 1

3. The ALC utility automatically displays the first OPEN-*x* FMT volume (in alphanumeric order) in the **VOLSER** field. If this is not the desired volume, select the desired volume from the drop-down list of volsers. If ALC could not find any OPEN-*x* FMT volumes, ALC displays the **DE format disk not found** message.
4. Enter the name of the dataset being allocated in the **Dataset** field (maximum forty-four characters: A-Z, 0-9, @, #, ., \). Do not use any spaces or symbols other than @, #, ., and \.
5. Enter or select the size of the new dataset (number of cylinders, number of tracks) in the **Cylinder** and **Track** fields. The file size will be (# of cyl) + (# of tracks). The **Max.** button enters the maximum size for the

new dataset in the **Cylinder** and **Track** fields based on the available capacity. The **Available Capacity** box displays the free space on the specified volume, so that you can select the appropriate size for the new dataset.

6. Enter or select the record format in the **Record format** field: F, FB, V, or VB.
7. Enter or select the record length in the **Record length** field:

   When record format = F, record length = block length.

   When record format = FB, record length = block length ÷ N (N = integer).

   When record format = V or VB, 5 ≤ record length ≤ (block length – 4).
8. Enter or select the block length in the **Block length** field. If block length = record length, select the **Copy** button to copy the record length into the **Block length** field.

   When record format = F or FB, 1 ≤ block length ≤ 32760.

   When record format = V or VB, 9 ≤ block length ≤ 32760
9. Select **Start** when all parameters for the new dataset are correct.
10. When the allocate operation completes successfully, the **Allocation complete** message is displayed. If the allocate operation could not be started due to an error condition, the **Allocate check error** message is displayed. If the allocate operation did not complete successfully, one of the following error messages is displayed (n = system error code):

Open error! (n)	Open process error on the OTO volume.
Seek error! (n)	Seek process error on the OTO volume.
Read error! (n)	Read process error on the OTO volume.
Write error! (n)	Write process error on the OTO volume.
Close error! (n)	Close process error on the OTO volume.

11. When you are finished allocating datasets on OTO volumes, select **Close** to close the Allocation panel and exit the ALC utility.

Creating FCU parameter definition files

The FCU parameter definition files are used to store FCU initiation parameter sets for pre-defined DE operations. If you plan to perform specific sets of DE operations more than once, you can create an FCU parameter definition file for each set of DE operations. The FCU program allows you to specify which FCU parameter definition file to load and execute, and you can also choose whether to execute all DE operations defined in the file or confirm each operation before starting it. You can create and edit the FCU parameter definition files interactively using the FCU GUI or manually using a text editor.

The table below lists the requirements for the FCU parameter definition files. Each set of FCU initiation parameters specifies the direction, source and target files, and FCU options (e.g., padding, delimiters) for a specific DE operation.

FCU Parameter Definition File Requirements

Item	**Requirement**
Default file name/location	For UNIX systems: **fcudata.param** in the directory containing the FCU program. For Windows 2000/2003/NT systems: **fcudata.prm** in the directory containing the FCU program. ***Note:*** When upgrading from version 01-01-24 or earlier to version 01-01-36 or later on Windows 2000/2003/NT systems, rename the file to change **.param** to **.prm**. To access the default FCU parameter definition file, leave the **param** option blank when starting FCU. To access a different parameter definition file, specify the file name (with complete path if necessary) when starting FCU. For Windows 2000/2003/NT systems, the FCU parameter definition file must have the **.prm** file extension.
Maximum number of parameter sets	For UNIX systems: 999. For Windows 2000/2003/NT systems: determined only by system memory.
FCU initiation parameters	Must be separated by at least one space character. Case sensitive (e.g., use **Yes**, not **YES** or **yes**, for the **Emp** and **RDW** parameters). Parameters (1)-(6) must be specified in order. Parameters (7)-(9) (optional) can be specified in any order (not before 1-6).

FCU Parameter Definition File Requirements	
Item	**Requirement**
Comment lines (start with #) When using the FCU GUI	For UNIX systems: cannot be created using GUI. loaded but not displayed by GUI ("Parameter file:Comment line" displayed). can be changed to a normal parameter line using GUI. included in number of parameter sets (each comment counts as one set). For Windows 2000/2003/NT systems: skipped by FCU (not processed). cannot be created, edited, or deleted using GUI (must use text editor). cannot be displayed by GUI ("Parameter file: Comment line" is displayed). not included in line count.
Comment lines (start with #) When not using the FCU GUI	For UNIX only: skipped by FCU (not processed). not included in number of parameter sets (max = 999).
Space lines	Not allowed with FCU GUI (remove space lines if upgrading from 01-01-24 or earlier to 01-01-36 or later on Windows 2000/2003/NT systems system). Allowed and skipped when using FCU without the GUI (using the **-nw** option).

The example below illustrates the structure of an FCU parameter definition file.

To define OTO operations, first define the OTM operations that transfer the data from the source files to the intermediate datasets, and then define the MTO operations, which transfer the data from the intermediate datasets to the target files. Do not define OTO operations that access OPEN-x FMT volumes and DE operations that access -A, -B, or -C DE volumes in the same FCU parameter definition file. Since FCU can only access one DE volume definition file at a time, each FCU parameter definition file must contain either DE operations which access OPEN-x FMT volumes or DE operations which access -A, -B, or -C volumes, but not both.

Example - FCU Parameter Definition File

```
mto   VSN:dataset name        Open system file name   CC   PAD   DEL   Emp=Yes   RDW=Yes
VSE=RF,RL,BL

mto   VSN:dataset name        Open system file name   CC   PAD   DEL   Emp=Yes   RDW=Yes
VSE=RF,RL,BL

otm   Open system file name   VSN:dataset name        CC   PAD   DEL   Emp=Yes
VSE=RF,RL,BL

otm   Open system file name   VSN:dataset name        CC   PAD   DEL   Emp=Yes
VSE=RF,RL,BL

 :

end
```

FCU Initiation Parameter Requirements

Number	Name	Function	Options
(1)	DE direction	Required. Specifies the file transfer direction.	**MTO** for DE mainframe-to-open. **OTM** for DE open-to-mainframe.
(2)	Source file	Required. Specifies the source file.	**VSN:dataset** for mainframe source dataset: VSN = six-character volume serial number colon = separates VSN and dataset name dataset = dataset name (44 chars max, no spaces) **file name** for open system source file: Space characters not allowed. Specify path (absolute or relative) if not in current directory: UNIX path: /directory_name/…/file_name Windows 2000/2003/NT systems path: drive:\directory_name\…\file_name
(3)	Target file	Required. Specifies the target file.	Same as parameter (2): **VSN:dataset** or **file_name**.
(4)	Code conversion	Required. Specifies code conversion.	**EA** to use default EBCDIC-ASCII code conversion table for MTO or OTM. **EcA** to use default EBCDIC-ASCII code conversion table for OTO. **No** for no code conversion. **File_name** to use your own code conversion table. (see ***Note 1***)

FCU Initiation Parameter Requirements			
Number	**Name**	**Function**	**Options**
(5)	Padding	Required. Specifies padding option.	**Yes** for MTO with padding. **No** for MTO without padding and for OTM. ***Note:*** When DE is used with DE Code Converter, padding in FCU is not supported. Specify “No.” Specifying “YES” results in an error.
(6)	Delimiter	Required. Specifies delimiter option and type.	For MTO operations: UNIX: **CR** Carriage return (CR) is added as a delimiter. **LF**Line feed (LF) is added as a delimiter. **No** No delimiter is added. Windows 2000/2003/NT systems: **CRLF** = CR + LF is added as a delimiter. For OTM operations: UNIX: **CR** Data up to CR is cut off as data entity. **LF**Data up to LF is cut off as data entity. **No** Data is cut off according to dataset record length. Windows 2000/2003/NT systems: **CRLF**Data up to CR + LF is cut off as data entity. **No** Data is cut off according to dataset record length. ***Note:*** When using with Code Converter, specify “No” because the delimiter for this parameter is not supported. Specifying other than “No” ends in an error in UNIX systems. In Windows systems, however, it is processed as “No.”
(7)	Empty file	Optional. Enables processing of empty source files.	This parameter is optional. If not specified, **Emp=No** is assumed. **Emp=Yes**Execute the data transfer even if the source file is empty. MTO target file size = 0. OTM target dataset will contain only EOF. **Emp=No**Execute the data transfer. If the source file is empty, the DE operation will be rejected with an error.

FCU Initiation Parameter Requirements			
Number	**Name**	**Function**	**Options**
(8)	Record description word	Optional. Specifies if FCU adds the record description word. MTO only. ***WARNING:*** Data transferred to open using RDW=Yes and then transferred back to S/390 is not compatible with the original dataset.	This parameter is optional. If not specified, **RDW=No** is assumed. Do not specify this parameter for OTM operations. **RDW=Yes**Add record description word to each record. CC, PAD, and DEL must be **No** (if not, error). Direction must be **MTO** (if not, error). Source dataset must be variable length (if not, **RDW=Yes** is ignored). **RDW=No**Do not add record description word to each record. Outputs only the data entity for each record. ***Note:*** When using with DE Code Converter, do not specify "RDW=Yes" because adding record length is not supported. Specifying "RDW=Yes" results in an error in UNIX systems, but it is processed as "No" in Windows NT systems.
(9)	VSE record	Optional. Specifies the RF, RL, and BL for VSE datasets. MTO and OTM only.	This parameter is optional. If not specified, the VTOC must specify the RF, RL, and BL. Do not specify this parameter for OTO using ALC-generated datasets on OPEN-x FMT volumes. **VSE=RF,RL,BL** RF, RL, and BL must be separated by a comma (**,**) and no spaces. **RF** **F** fixed-length and unblocking **FB** fixed-length and blocking **V** variable-length and unblocking **VB** variable-length and blocking **RL** record length in bytes (decimal) When RF = F: RL = BL When RF = FB: RL = [BL/n] (n is an integer) When RF = V or VB: 5 ≤ RL ≤ [BL - 4] **BL**When RF = F or FB: 1 through 32760 When RF = V or VB: 9 through 32760 (See ***Note 2***)
(10)	Carriage return	Required. Marks end of parameter set.	Press **Return** (**Enter** for Windows NT2000/2003 systems) at the end of each line. Note: When using DE with DE with Code Converter, specify 'USER-EDIT field definition file name, edit option file name' between (10) and (11). For details, see the *DE Code Converter User's Guide*, MK-93RD152-P.

FCU Initiation Parameter Requirements			
Number	**Name**	**Function**	**Options**
(11)	End of file	Optional. Marks end of parameter file.	**end**This parameter is optional.

Note 1: The Code converting function is not supported for OTO. Specify "No" in this field when OTO is used. Even if "EA" is specified or the file name of the conversion table in this field is specified when OTO is used, code conversion is not executed. Specifying anything other than "No" results in an error in UNIX systems. In Windows systems, specifying anything other than "No" is processed as "No."

Note 2: When you use MTO and OTM for a dataset allocated by VSE 2.5, Data Exchange can transfer data without the VSE parameter. This is illustrated below:

RF=V: It is possible to transfer data between correct dataset attributes

(**5** ≤ RL ≤ **BL** ≤ **4**). The data transfer is valid **only** if the VSE parameter is the following:

RL ≤ 32756

BL ≤ 32760

BL = RL+4

For the dataset attribute below, the data transfer is valid **only** if user specifies the VSE parameter value as shown above.

RL ≤ 32756

BL ≤ 32760

For the following dataset attribute, the data transfer is valid **only** if user specifies the VSE parameter value between RL and BL values shown as #1 and #2 below.

RL ≤ 32756

BL ≤ 32760

RL = BL

#1: RL(Input value for VSE parameter) = RL(value on VTOC) + 4 $\leq$ 32756

#2: BL(Input value for VSE parameter) = BL(value on VTOC) + 8 $\leq$ 32760

- RF=VB: It is possible to transfer data using the correct dataset attribute ($5 \leq RL \leq BL-4$). The data transfer is valid **only** if VSE parameter is the following value:

  RL(Input value for VSE parameter) = RL(value on VTOC) + 4 $\leq$ 32756

  BL(Input value for VSE parameter) = BL(value on VTOC) + 8 $\leq$ 32760

- RF=V (without VSE parameter): It is possible to transfer data using the correct dataset attributes ($BL \leq RL+4 \leq 32760$). The data transfer is valid **only** if the RL and BL values on VTOC are the following:

  $BL \leq RL+4 \leq 32760$

  Where $RL > 32756$ and $BL > 32760$ on VTOC, DE manages the data as RL=32756 and BL=32760.

  Where $RL \leq 32756$, $BL \leq 2760$ and RL = BL on VTOC, DE manages the data as shown:

  RL(Data Exchange internal value) = RL(value on VTOC) + 4 $\leq$ 32756

  BL(Data Exchange internal value) = BL(value on VTOC) + 8 $\leq$ 32760

- RF=VB (without VSE parameter): It is possible to transfer data between correct dataset attributes ($BL = RL+4 \leq 32760$). The data transfer is valid **only** if the RL and BL values on VTOC are the following:

  $BL \leq RL+4 \leq 32760$

  Where $RL \leq BL \leq 32752$ on VTOC, DE manages the data as shown:

  RL(Data Exchange internal value) = RL(value on VTOC) + 4

  BL(Data Exchange internal value) = BL(value on VTOC) + 8

Multiple volume definition file

The Multiple Volume Dataset function is supported for MTO operations.The name of the multiple volume definition file is ***"multidef.dat***". This file must be placed under the current directory where the DE is to be executed.

Specified the other name in "FAL_MULTI_DEF_FILE" of the environment variable, if the name of the multiple volume definition file want to be changed.

VSN Function

```
VSN:DSN[,VOLID1] VSN[,VOLID2]  -----  VSN[,VOLIDn]
end
```

This function is defined below:

- **VSN:DSN {,VOLID1}**

  This parameter is the information of the head volume.

  - **VSN**: a volume serial number with six digit of alphabet (A-Z, @, #, and \) or numeral (0-9) characters.
  - **DSN**: dataset name. (Use maximum 44-digit of alphabet or numeral character)
  - **VOLID1**: a VSN identification (Omit this parameter if a VSN identification is omitted in the volume definition file, and specify same as the volume definition file, if a VSN identification is specified in the volume definition file.)

- **VSN[,VOLID2]**

  This parameter is the information of the second volume:

  - **VSN**: a volume serial number with six digit of alphabet (A-Z, @, #, and \) or numeral (0-9) characters.
  - **VOLID2**: a VSN identification (Omit this parameter if a VSN identification is omitted in the volume definition file, and specify

same as the volume definition file, if a VSN identification is specified in the volume definition file.)

- **VSN[,VOLIDn]**

  This parameter is the information of the last volume. (The number of volume is ‘n’):

  - **VSN**: a volume serial number with six digit of alphabet (A-Z, @, #, and \) or numeral (0-9) characters.
  - **VOLIDn**: a VSN identification (Omit this parameter if a VSN identification is omitted in the volume definition file, and specify same as the volume definition file, if a VSN identification is specified in the volume definition file.)
- Each line above must be separated by using ***“Return”*** key.
- The ***“end”*** specifies that the volume definition file ends here.

Note: Each parameter must be separated with one or more ***“space”*** character.

Note: One data set information must be specified in one line.

Note: 999 information can be specified in the multiple volume definition file.

5

FCU file transfer operations

DE file transfer operations are performed using the FCU GUI software installed on the open system host attached to the disk array.

The FCU GUI enables you to perform file transfer operations interactively, provides access to detailed information on the DE source datasets and files, and displays error information for DE operations. The FCU GUI also allows you to create and modify FCU parameter definition files interactively.

When you perform DE operations that access datasets on -A, -B, or -C DE volumes, FCU must have access to the DE volume definition file that defines these volumes. When you perform DE operations that access ALC-generated datasets on OPEN-*x* FMT DE volumes, FCU must have access to the separate OTO volume definition file that defines the OPEN-*x* FMT volumes.

Since FCU can only access one DE volume definition file at a time, the FCU parameter definition files must also keep operations using OPEN-*x* FMT volumes separate from operations using -A, -B, or -C DE volumes.

Before you start FCU GUI operations, make sure that the desired DE volume definition file is available (**datasetmount.dat** in current directory) and that the desired FCU parameter definition file contains DE operations which access the volumes defined in the DE volume definition file. FCU will not be able to perform operations which access volumes that are not defined in the current DE volume definition file.

For information on using the FAL C functions (Visual C++ for Windows NT/2000/2003 systems), which enable user programs on the open system host to access S/390, datasets on DE volumes, see "Using FCU without the GUI" (page 195).

Using the FCU GUI for UNIX

The FCU GUI enables you to perform DE file transfer operations interactively and provides access to detailed information on the datasets/files in the specified DE source volume/directory. The FCU GUI displays the DE operations in the FCU parameter definition file (if specified), allows you to modify the FCU parameter definition file interactively, and also allows you to enter FCU parameters and perform DE operations manually. The FCU GUI also displays the error information for DE operations.

Starting the FCU GUI for UNIX

To start the FCU GUI program for UNIX -based platforms:

1. At the UNIX command line prompt, enter: **fcu [-nc] [param]**

   The **-nc** option (**nc** = no checking) tells FCU to execute all specified DE operations without requesting confirmation for FCU parameters or checking for existing MTO target files. If you want to bypass these confirmations, enter **-nc**.

   The **param** option tells FCU whether to use the FCU parameter definition file or a specific FCU initiation parameter set to perform DE operations. The **param** option must have one of the following three values:

   - [blank]. If you want to use the default FCU parameter definition file (**fcudata.param** in the current directory), leave the **param** option blank (do not enter anything).
   - **file_name**. If you want to use a different FCU parameter definition file, enter the file name with complete path (absolute or relative) if not in the current directory.
   - **-P + parameters**. If you want to perform one specific DE operation, enter **-P** followed by the FCU initiation parameter set (e.g., **mto VSN:dataset targetfile No No No**) for the desired DE operation. The **-P** option requires the **-nc** option.

For example:

- If you want to use the default FCU parameter definition file and check the parameters and MTO target files, enter: **fcu**
- If you want to use the default FCU parameter definition file and perform all operations without checking parameters or MTO target files, enter: **fcu -nc**
- If you want to use a different FCU parameter definition file and perform all operations without checking parameters or MTO target files, enter: **fcu -nc filename**
- If you want to perform one specific DE operation, enter: **fcu -nc -P [parameters]**

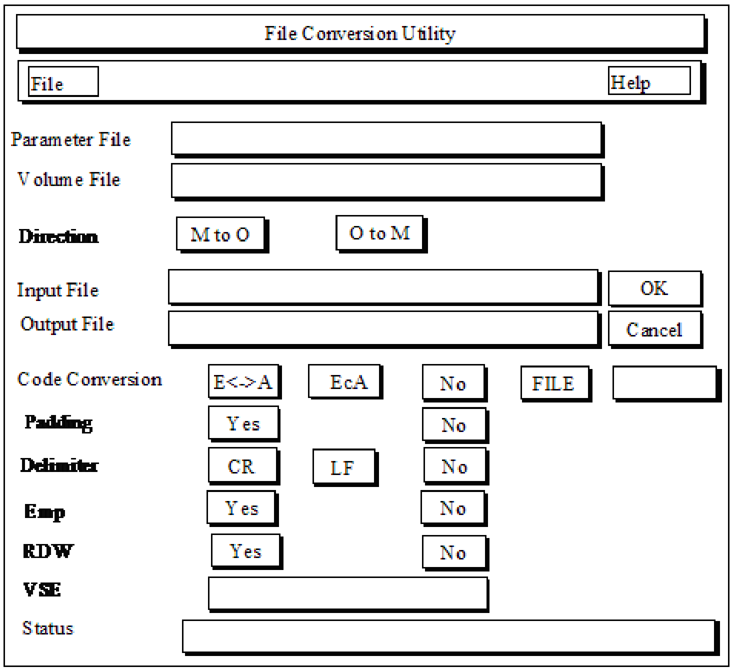
Note: The following warnings may appear during FCU startup. These warnings do not affect FCU and can be ignored.

- ***WARNING:*** Missing characters in String to FontSet conversion.
- ***WARNING:*** Cannot convert string "-dt-interface system-medium-r-normal -m*-*-*-*-*-*-*-*-*" to type FontSet.

2. The FCU GUI program now starts loading. The FCU version and copyright screen is displayed while FCU is loading. When FCU is finished loading, the FCU main window is displayed.
3. If you specified the **-nc** option, FCU processes all specified operations, overwrites existing MTO target files, terminates, and displays any error information at the UNIX prompt.

FCU main window for UNIX

The FCU main window displays the FCU initiation parameter sets in the specified FCU parameter definition file (if available), allows you to perform DE operations, and provides access to all FCU functions.



- **Parameter File**. The **Parameter File** field displays the FCU parameter definition file that you specified by the **param** option when you started FCU. If this field is blank, FCU could not find the default or specified FCU parameter definition file. If you want to use an FCU parameter definition file, you can enter the desired file name in this field (complete path if not in the current directory). If you do not want to use an FCU parameter definition file, you can leave the **Parameter File** field blank and enter the FCU initiation parameters manually.
- **Volume File**. The **Volume File** field displays the DE volume definition file. This file must have the default name and location (**datasetmount.dat** in current directory). If this field is blank, FCU could not find the file and will not be able to perform DE operations. In this case, exit FCU, and create the DE volume definition file.

  When FCU starts up, the first set of FCU initiation parameters is automatically loaded from the specified FCU parameter definition file (unless the file is not found). If desired, you can change any of the parameters, or you can use the **File-Load** command to load the next parameter set. The FCU initiation parameters are:

- **Direction**. The **Direction** buttons allow you to select the desired direction for the DE operation: **M to O** = MTO, **O to M** = OTM.
- **Input File**. The **Input File** field allows you to enter the name of the DE source file. For MTO, enter the S/390 volser and dataset name (**VSN:dataset**). For OTM, enter the UNIX file name (with complete path if not in the current directory).
- **Output File**. The **Output File** field allows you to enter the name of the DE target file. For MTO, enter the UNIX file name (with complete path if not in the current directory). For OTM, enter the VSN and dataset name (**volser:dataset**).
- **Code Conversion**. The **Code Conversion** buttons allow you to select the desired code conversion option: **E<->A** = default code conversion table, **EcA** = default code conversion table (for OTO only), **No** = no code conversion, **File** = enter the file name of your conversion table (with complete path if not in current directory).
- **Padding**. The **Padding** buttons allow you to select the desired padding option: **Yes** = padding, **No** = no padding.
- **Delimiter**. The **Delimiter** buttons allow you to select the desired delimiter option: **CR** = carriage return, **LF =** line feed, **No =** no delimiters.
- **Emp**. The **Emp** buttons allow you to select the empty file option: **Yes** = source file is empty, **No** = source file is not empty.
- **RDW**. The **RDW** buttons allow you to select the record description word option (MTO only): **Yes** = add RDW to each record (**Code Conversion**, **Padding**, and **Delimiter** must be **No**), **No** = do not add RDW to each record.
- **VSE.** The VSE field allows you to enter the VSE record information: **RF,RL,BL**

  Use a comma (no spaces) between each value.
- **OK**. The **OK** button starts the specified DE operation. The **Cancel** button removes the values entered and returns the FCU main window to the initial settings. (The **Cancel** button does not cancel the DE

operation in progress.) Be careful not to click **OK** or **Cancel** more than once.

- The **Status** field displays the status of the requested DE operation:
  - **Now checking** = FCU is executing a dataset search or file attribute check. If you specified the **-nc** option when you started FCU, this check does not occur.
  - **Overwrite? (OK/Cancel)** is displayed if the MTO target file already exists. Click **OK** to overwrite the existing file, or click **Cancel** to cancel the requested operation. If you specified the **-nc** option when you started FCU, this confirmation does not occur.
  - **x%** = The requested DE operation is *x*% complete.
  - **Complete** = The requested DE operation completed successfully.
  - **Error**. The **Status** field also displays error information for FCU and DE operations. See "Troubleshooting" (page 167) for further information on error conditions.

File menu commands

The **File** menu provides access to the following FCU functions:

- **Load**. This command loads the parameter sets from the specified FCU parameter definition file onto the FCU main window. Each time you select **Load**, the next set of parameters is loaded. If you enter a file name in the **Parameter File** field, the **Load** command opens the file and loads the first parameter set (or creates the new file). If the FCU parameter definition file is empty or was not found, FCU ignores this command.
- **Save**. This command saves the FCU parameter definition file. If no FCU parameter set was previously loaded, the current parameter set is added to the file. If a parameter set was previously loaded and you made changes, the current parameter set overwrites and replaces the previously loaded parameter set. If you make changes and do not select **Save**, FCU will discard your changes when you select **Load** or **Exit**.
- **Delete**. This command deletes the currently loaded parameter set from the FCU parameter definition file. If the FCU parameter definition file

does not yet exist or does not contain the parameter set on screen, FCU ignores this command.

- **Exit**. This command closes the current FCU parameter definition file (unsaved changes are discarded), and then closes the FCU program.

Help menu commands

The **Help** menu provides access to the following FCU functions. ***Note:*** When installing Code Converter, the Edit_prm menu is displayed and the parameters for DE Code Converter can be displayed.

- **Volume**. This command displays the contents of the DE volume definition file, so that you can verify that the DE volumes are properly defined.
- **MF-File**. This command displays the dataset information for each dataset in the specified mainframe (MF) volume. The VSN must be entered in the **Input File** field (for MTO) or **Output File** field (for OTM) on the FCU main window.
    - **Dataset name**: An asterisk (*) before the dataset name indicates that DE can process the dataset. A dash (-) indicates that DE cannot process the dataset. A question mark (?) indicates that FCU can process the dataset only if the VSE record option is used to specify the RF, RL, and BL.
    - **Dataset organization (DO) type**: SAM, DAM, PAM, VSAM, ??? = unknown.
    - **Record format (RF):** F = fixed length, V = variable length, U = undefined length, S = spanned record, ? = unknown.
    - **Block length (BL)**: in bytes
    - **Record length (RL)**: in bytes
    - **Dataset size (DS)**: in tracks
- **UX-File**. This command displays the UNIX (UX) files in the directory specified in the **Input** or **Output File** field on the FCU main window. If no directory is specified in the **Input File** or **Output File** field, FCU displays the files in the current directory. If a nonexistent directory is specified, FCU will return an error.

- **Error**. This command opens the error information window, which displays the FAL, FCU, and system error codes/messages (see 5.6).
- **OnVersion**. This command displays the FCU version and copyright information screen.

```
/dev/rdsk/cxtydz  volser  3390-3B
/dev/rdsk/cxtydz  volser  3390-3A
/dev/rdsk/cxtydz  volser  3390-3C
/dev/rdsk/cxtydz  volser  3380-KB
/dev/rdsk/cxtydz  volser  3380-KA
/dev/rdsk/cxtydz  volser  3380-KC
 :
 :
end
```

Help MF-File display

```
Dataset Information : VSN = xxxxxx   Device Emulation Type = 3390-3B

Dataset Name           DO       RF     BL       RL       DS


*SAMFILE01.FIX         SAM      F      4096     128      150 ← Can be processed by FCU.

-DAMFILE.F             DAM      F      4096     128      30 ← Cannot be processed by FCU.

*SAMFILE02.VAR         SAM      V      4000     80       50 ← Can be processed by FCU.

-PAMFILE               PAM      F      5000     100      200 ← Cannot be processed by FCU.

-VIRTUALSTORAGEACCESS  VSAM     V      32768    4096     3000 ← Cannot be processed by FCU.

-UNDEFSAMFILE          SAM      U      8000     200      80 ← Cannot be processed by FCU.

-SAMFILESPANNED        SAM      S      8192     8192     300 ← Cannot be processed by FCU.
```

Help UX-File display

```
UNIX FILE LIST : DIR = /aaaaa/bbbbb/cccc
ddddd.dd       eeeeee       ffffff.fffffff
hhhh.hhhh     zzzzzz.z     xxxx.x
yyyyyyyyy
```

Creating an FCU parameter definition file using FCU for UNIX

To create an FCU parameter definition file using the FCU GUI for UNIX:

1. Start the FCU GUI for UNIX by entering **fcu**. Do not specify the **-nw**, **-nc**, or **param** option.
2. If you plan to perform DE operations while you are creating the FCU parameter definition file, make sure that the **Volume File** field displays the correct DE volume definition file (**datasetmount.dat**). If not (or if incorrect), FCU will not be able to perform DE operations, but you can still create a new FCU parameter definition file. When the FCU main window opens, enter the desired file name in the **Parameter File** field.
3. Select the **File-Load** command to open the new file.
4. Enter the desired FCU initiation parameters for the first DE operation:
   a) Select the file transfer direction using the **M to O** button or **O to M** button.
   b) Enter the source and target files in the **Input File** and **Output File** fields (**VSN:dataset**, **filename** with complete path if not in current directory).
   c) Select the desired FCU file transfer options: **Code Conversion**, **Padding**, **Delimiter**, **Emp**, **RDW**, and **VSE**.
5. When the FCU initiation parameters are correct, select the **File-Save** command to add this parameter set as the first line in the new FCU parameter definition file. If the DE volume definition file is correct, you can perform the operation now by clicking **OK**. If the **OK** button is not enabled, the parameter set has not been saved in the file.
6. Select the **File-Load** command to load the next line. The **Status** field should indicate that you are at the end of the file. The FCU GUI for UNIX only allows you to add new lines when you are at the end of the file (right after the last line).
7. Repeat steps (5), (6), and (7) to add each parameter set to the new FCU parameter definition file. Make sure to keep DE operations which use OPEN-*x* FMT volumes in a separate FCU parameter definition file from operations which use -A, -B, -C volumes.

8. If you need to modify an existing line, go to the line to be modified using the **File-Load** command, change the parameters as needed, and then use the **File-Save** command to replace the line that was loaded.
9. If you need to insert a new line between existing lines, use a text editor later to edit the file. You cannot add a new line between existing lines using the FCU GUI for UNIX.
10. When you are finished adding lines to your new FCU parameter definition file, make sure that you have selected the **File-Save** command for the last parameter set you added or modified, and then select the **File-Exit** menu command to close the file and exit FCU.

Performing DE operations using the FCU GUI for UNIX

To perform DE file transfer operations using the FCU GUI for UNIX:

1. If you will be performing MTO operations:
   a) Make sure that the source datasets are located on the desired DE volume(s). If you will not be using an existing FCU parameter definition file, write down the VSN:dataset of the source dataset and the complete path and file name of the target file for each MTO operation.

   b) Verify that the MTO target files do not already exist (or can be overwritten).

   c) Vary the MTO volume(s) and channel path(s) offline from the S/390 host.
2. If you will be performing OTM operations:
   a) Make sure that the source files are located on the desired DE volume(s). If you will not be using an existing FCU parameter definition file, write down the complete path and file name of the source file and the VSN: dataset of the target dataset for each OTM operation.

   b) Create and allocate the target datasets. This ensures that the target dataset is registered in the VTOC. Make sure to allocate enough space and to use the appropriate record format and record length for the data to be transferred.

   c) Vary the OTM volume(s) and channel path(s) offline from the S/390 host.

3. If you will be performing OTO operations:
   a) If you will not be using an existing FCU parameter def. file, write down the complete path and file name of the source and target files for each DE otm/mto operation.
   b) Allocate the intermediate datasets on the OTO volume(s). Use the ALC utility on OPEN-x FMT volumes. Make sure to allocate enough space and to use the appropriate record format and record length for the data to be transferred.
   c) Verify that the OTM target files do not already exist (or can be overwritten).
4. Make sure that the desired DE volume definition file (OTM only, or MTO and OTM) is available for use by FCU (**datasetmount.dat** in current directory).
5. Start FCU with the desired options. ***Note:*** If you specify the **-nc** option, FCU performs all specified operations continuously, then terminates and displays any error information at the UNIX prompt.
6. When the FCU main window opens, make sure that the **Volume File** field displays the DE volume definition file (**datasetmount.dat**). If the DE volume definition file is not displayed (or if incorrect), FCU will not be able to perform DE operations.
7. Make sure that the **Parameter File** field displays the desired FCU parameter definition file. If not, enter the desired FCU parameter definition file name (with complete path if not in the current directory), and select the **File-Load** command to open the file. If you want to create a new file using the FCU GUI, see "Creating an FCU parameter definition file using FCU for Windows" (page 123) for instructions.
8. The FCU main window displays the first/next parameter set in the specified FCU parameter definition file. If you want to perform this DE operation, click **OK**. If not:
   a) You can load the next parameter set using the **File-Load** command.

   b) You can delete the current parameter set from the FCU parameter definition file using the **File-Delete** command. The next parameter set loads automatically.

   c) You can modify the current parameter set as follows: change the FCU parameters as needed, and then use the **File-Save** command to save your changes in the FCU parameter definition file (replaces the previously loaded parameter set).

   d) You can add a new parameter set to the end of the file as follows: select **File-Load** until you reach the end of the file, enter the desired parameters, and then select **File-Save** to add the new line at the end. If you want to insert a new line between existing lines, edit the FCU parameter definition file later using a text editor.

9. When the desired DE operation is displayed, click **OK** to start the operation. (If the **OK** button is not enabled, you have not saved the current parameter set.)
10. If you started an MTO operation and the target file already exists, FCU requests overwrite confirmation. Click **OK** to overwrite the target file, or click **Cancel** to cancel the operation.
11. When FCU starts the operation, the **Status** field displays the progress of the operation. If desired, while the operation is in progress, you can load another parameter set and click **OK** to start the next operation right after the current operation completes. ***Note:*** Be careful when doing this. If you click buttons or menu commands while an operation is in progress, FCU will save and execute those commands when the current operation completes.
12. When the DE operation is complete, the **Status** field displays **Complete**. If an error occurred, the error information display opens automatically to display the error. See "Troubleshooting" (page 167) for further information on error conditions.
13. FCU does not load the next operation automatically. To perform another DE operation, select **File-Load**, and repeat steps (8) through (12). To exit FCU, select the **File-Exit** command.

Using the FCU GUI for Windows NT/2000/2003

Starting the FCU GUI for Windows NT/2000/2003

To start the FCU GUI program for Windows 2000/2003/Windows NT systems:

1. Log on with Administrator access privileges.
2. Select **Start-Programs-FCU-FCU**. If you want to specify any of the FCU option, start FCU from the command line as explained in step 3.

   Note: Do not start FCU by dragging and dropping an FCU parameter definition file on the FCU program icon. FCU program operation cannot be guaranteed.

3. If you want to specify any of the FCU options, start FCU from the command line (DOS prompt) as follows: go to the FCU directory (containing **fcu.exe** and **datasetmount.dat**), and enter **fcu [-nc] [-cl] [param]**

   The **-nc** option is the same as for UNIX: all specified DE operations are performed without confirmation of FCU parameters or MTO target file overwrites.

   The **-cl** option specifies that all FCU log files will be cleared before starting.

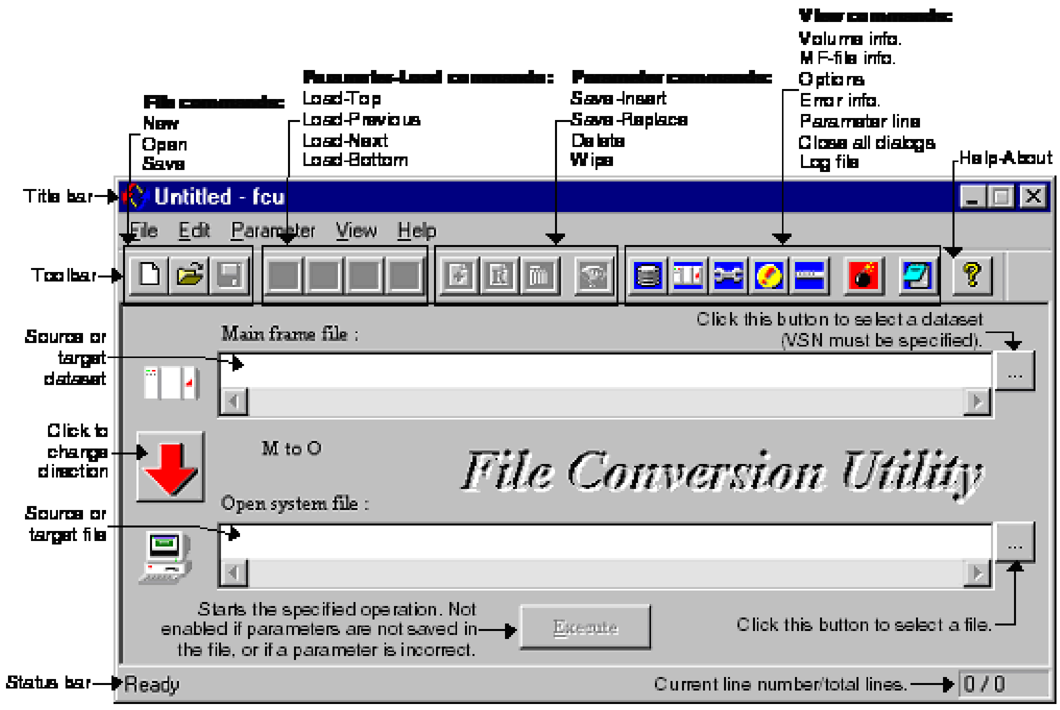
   The **param** option is the same as for UNIX:

   - If you want to open a new untitled FCU parameter definition file when you start FCU, leave the **param** option blank.
   - If you want to load an FCU parameter definition file when you start FCU, enter the file name with complete path if the file is not in the current directory.

4. If you started FCU from the DOS prompt and specified the **-nc** option, FCU processes all specified operations, overwrites existing MTO target files, and then terminates and displays any error information at the DOS prompt.

FCU main window for Windows NT/2000/2003

The FCU main window displays the FCU parameter definition file (or **Untitled** if no file was specified), allows you to perform DE operations, and provides access to all FCU functions.



Click this button to select a dataset (VSN must be specified).

Starts the specified operation. Not enabled if parameters are not saved in the file, or if a parameter is incorrect.

Click this button to select a file.

The FCU title bar displays the current FCU parameter definition file. The toolbar provides speed buttons for the commonly used FCU functions. The status bar displays the current line number and total number of lines in the current FCU parameter definition file. The **Main frame file** and **Open-system file** fields display the files to be transferred (no spaces allowed).

The **File** menu provides access to the following FCU functions:

- The **File-New** command (Ctrl+N) opens a new FCU parameter def. file (untitled.prm).

- The **File-Open** command (Ctrl+O) opens an existing FCU param. def. file (filename.prm).
- The **File-Save** command (Ctrl+S) saves the current FCU parameter definition file. Deleted and replaced lines are discarded, inserted lines are added, and all lines after **end** are discarded. ***Note:*** This command does not save the current parameter set.
- The **File-Save As…** command saves the current FCU parameter definition file with a different file name and/or location.
- The **File-Exit** command (Ctrl+X) exits the FCU software.

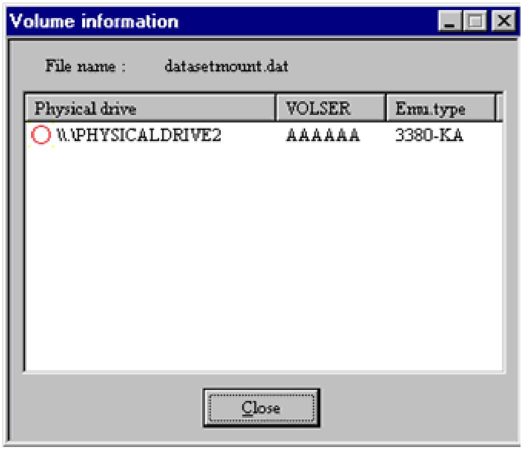
The **Edit** menu is reserved for future enhancement and is not yet enabled.

The **Parameter** menu provides access to the following FCU functions:

- The **Parameter-Load** command loads the **Previous**, **Next**, **Top**, and **Bottom** parameter lines from the current FCU parameter definition file. The FCU main window status bar updates the current line number when any **Parameter-Load** command is executed.
- The **Parameter-Save** command allows you to either **Insert** the parameter set being displayed into the current FCU parameter definition file, or **Replace** the current parameter set (previously loaded) with the parameter set being displayed. If you do not select this command, your parameter changes will not be saved. ***Note:*** This command does not save the current FCU parameter definition file (you must use **File-Save/Save As**).
- The **Parameter-Delete** command deletes the current parameter set from the current FCU parameter definition file. The line is not permanently deleted until you save the current FCU parameter definition file using the **File-Save** command.
- The **Parameter-Wipe** command clears all FCU initiation parameters displayed on screen, so that you can input new parameters easily. This command does not delete the current parameter set.

The **View** menu provides access to the following FCU functions:

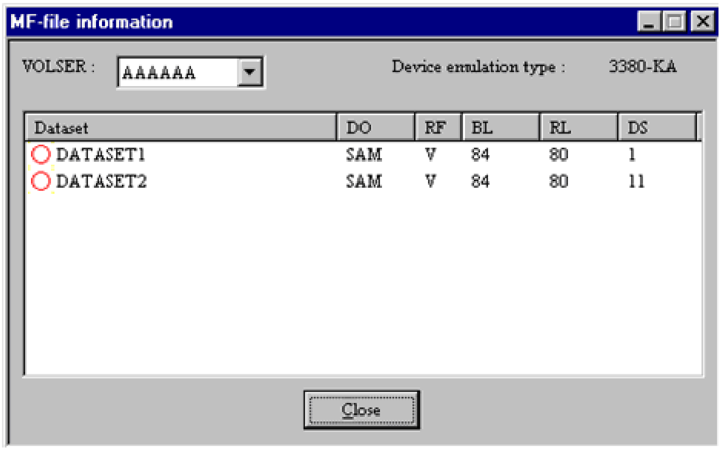
- The **View-Toolbar** and **View-Status bar** commands display/hide the FCU toolbar and status bar. The toolbar provides speed buttons for the commonly used FCU functions. The status bar displays the current line number and total number of lines in the current FCU parameter definition file.
- The **View-Volume information…** command opens the FCU Volume Information window, which displays the contents of the DE volume definition file. A O displayed next to a volume indicates that the volume definition is correct and FCU can access the volume. An **5** displayed next to a volume indicates that the volume definition is not correct and FCU cannot access the volume.



File name : datasetmount.dat

Physical drive	VOLSER	Emu.type
○ \\.\PHYSICALDRIVE2	AAAAAA	3380-KA

- The **View-MF-file information…** command displays the following information for the mainframe (MF) files (datasets) in the volume specified in the **Mainframe file** field on the FCU main window.



VOLSER : AAAAAA   Device emulation type : 3380-KA

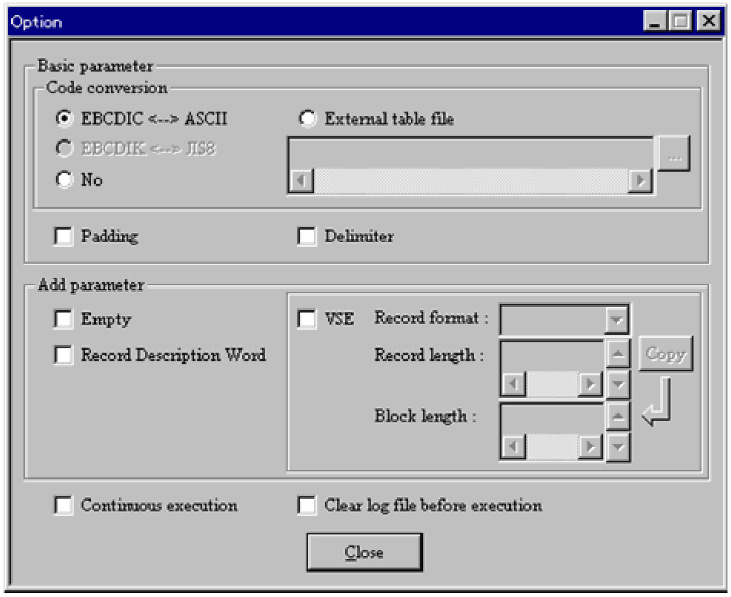
Dataset	DO	RF	BL	RL	DS
○ DATASET1	SAM	V	84	80	1
○ DATASET2	SAM	V	84	80	11

- ○ = The dataset can be processed by FCU.

  5 = The dataset cannot be processed by FCU.

  ? = The dataset can be processed by FCU only if the VSE record option is used to specify the RF, RL, and BL.
- **Dataset** = dataset name
- **DO** = dataset organization type: **SAM**, **DAM**, **PAM**, **VSAM**, **?** (other than above)
- **RF** = record format: **F** (fixed-length), **V** (variable-length), **U** (undefined), **S** (spanned), **?** (other than above)
- **BL** = block length
- **RL** = record length
- **DS** = dataset size (in number of tracks)

- The **View-Option…** command opens the Option window, which allows you to view/change the FCU file transfer options (code conversion, padding, delimiter, empty file, RDW, and VSE), continuous execution option, and clear log file option.



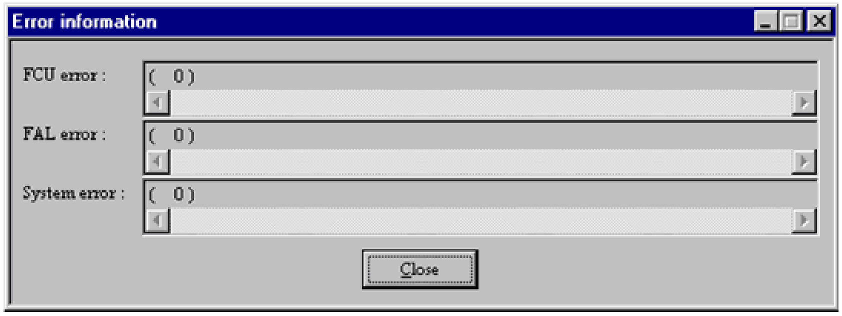
- Select the **EBCDIC<-->ASCII** button to use the default code conversion table (**EA**, **EcA**). Select **No** for no code conversion (**No**). Select **External table file** and enter the file name with path if not in the current directory (e.g., **/directory/filename.tbl**).
- Check the **Padding** box to request the padding option (**Yes**).
- Check the **Delimiter** box to request the delimiter option (**Yes**).
- Check the **Empty** box to request the empty file option (**Emp=Yes**).
- Check the **Record Description Word** box to request the RDW option (**RDW=Yes**).

  Note: If the **Record description word** box is checked, the **EBCDIC<-->ASCII** button and the **Padding** and **Delimiter** boxes are ignored.
- Check the VSE box and enter the RF, RL, and BL to request the VSE record option.
- Check the **Continuous execution** box to tell FCU to process the rest of the FCU initiation parameter sets in the specified FCU parameter definition file without stopping after each completed

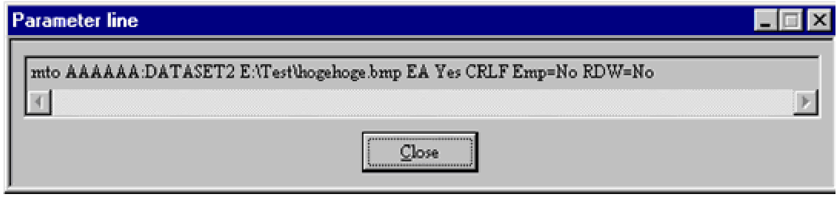
operation (equivalent to the **-nc** option starting at the desired line). FCU will execute all lines from the current line to the **end**. If you do not check the **Continuous execution** box, FCU will stop after each operation.

- Check the **Clear log file before execution** box to clear the log file for the current FCU parameter definition file (e.g., fcudata.prm.log). The user should clear the FCU log files as needed to decrease the file size and save space on the current drive.

- The **View-Error information…** command opens the Error Information window, which displays the most recent error information (error code and message) for FCU, FAL, and system errors.



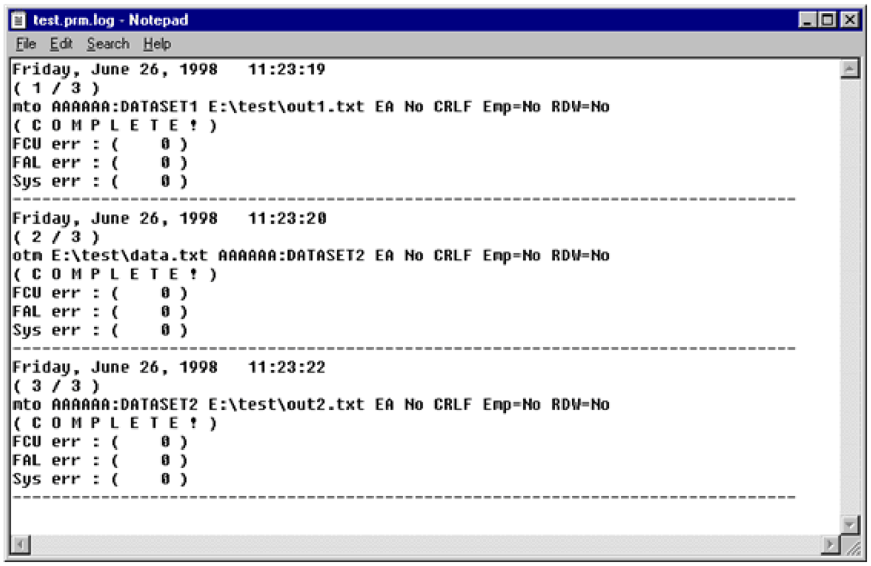
- The **View-Parameter line…** command opens the Parameter Line window, which displays the current line (parameter set) in the current FCU parameter definition file.



```
mto AAAAAA:DATASET2 E:\Test\hogehoge.bmp EA Yes CRLF Emp=No RDW=No
```

- The **View-Close all dialogs** command closes all open windows except the FCU main window.

- The **View-Log file** command opens the log file for the current FCU parameter definition file using the Windows Notepad text editor. The log file contains the parameter sets executed, the date and time of execution, the result of each operation, and the error information (FCU, FAL, and Sys error codes) for each operation.



```
Friday, June 26, 1998   11:23:19
( 1 / 3 )
mto AAAAAA:DATASET1 E:\test\out1.txt EA No CRLF Emp=No RDW=No
( C O M P L E T E ! )
FCU err : (    0 )
FAL err : (    0 )
Sys err : (    0 )
--------------------------------------------------------------------------------
Friday, June 26, 1998   11:23:20
( 2 / 3 )
otm E:\test\data.txt AAAAAA:DATASET2 EA No CRLF Emp=No RDW=No
( C O M P L E T E ! )
FCU err : (    0 )
FAL err : (    0 )
Sys err : (    0 )
--------------------------------------------------------------------------------
Friday, June 26, 1998   11:23:22
( 3 / 3 )
mto AAAAAA:DATASET2 E:\test\out2.txt EA No CRLF Emp=No RDW=No
( C O M P L E T E ! )
FCU err : (    0 )
FAL err : (    0 )
Sys err : (    0 )
--------------------------------------------------------------------------------
```

- The **Help-About FCU…** menu command opens the FCU version and copyright screen, so that you can check the FCU program version information.

Creating an FCU parameter definition file using FCU for Windows

To create an FCU parameter definition file using the FCU GUI for Windows 2000/2003/Windows NT systems:

1. Start the FCU GUI for Windows 2000/2003/Windows NT systems. If you start FCU from the DOS prompt, enter **fcu** (do not specify the **-nc** or **param** option).
2. When the FCU main window opens, the title bar should display **Untitled** to indicate that a new parameter definition file is open. If a file name is displayed instead of **Untitled**, use the **File-New** command to open a new parameter definition file.

3. If you plan to perform DE operations while you are creating the FCU parameter definition file, open the Volume information window (select **View-Volume information…**), and make sure that the desired DE volume(s) is/are available. If not, FCU will not be able to perform DE operations, but you can still create a new parameter definition file.
4. Enter the desired FCU initiation parameters for the first DE operation.
   a) Use the ⬆ ⬇ buttons to select the transfer direction (**M to O** or **O to M**).
   b) Enter the source and target datasets/files in the **Mainframe File** field (VSN:dataset), and/or **Open-system file** field (/directory/…/filename).
   c) Open the Option window using the **View-Option…** menu command, and then select the desired FCU options (code conversion, padding, delimiters, etc.). Do not select **Continuous execution** or **Clear log file** when creating a new FCU parameter definition file. Close the Option window when you are done.
5. When the FCU initiation parameters are correct, select the **Parameter-Save-Insert** command to save the current parameter set in the new FCU parameter definition file. The status bar now displays 1/1 to indicate that line one of one is now being displayed. If the DE volume is available, you can perform the operation now by clicking **Execute**. If the **Execute** button is not enabled, the parameter set has not been saved in the file.
6. If you need to insert a new line between two existing lines, go to the line above/before the line to be inserted using the **Parameter-Load** commands, change the parameters as needed, and then use the **Parameter-Save-Insert** command to insert the new line. The new line is inserted below/after the current line number.The **Cancel** button changes to **Close** after the operation is complete.Repeat steps (4) and (5) to add each new FCU initiation parameter set to the new FCU parameter definition file. If desired, you can use the **Parameter-Wipe** menu command to clear the screen before you enter the next set of parameters, or you can leave the parameters and make changes where needed to specify the next new line in the file. Make sure to keep DE

operations which use OPEN-*x* FMT volumes in a separate FCU parameter definition file from operations which use -A, -B, and -C volumes.

7. If you need to modify an existing line, go to the line to be modified using the **Parameter-Load** commands, change the parameters as needed, and then use the **Parameter-Save-Replace** command to modify the line as specified.
8. When you want to save your new FCU parameter definition file, select the **File-Save** menu command. The file extension must be **.prm**.

Performing DE operations using the FCU GUI for Windows

To perform DE file transfer operations using the FCU GUI for Windows 2000/2003/Windows NT systems:

1. If you will be performing MTO operations:
   a) Make sure that the source datasets are located on the desired DE volume(s). If you will not be using an existing FCU parameter definition file, write down the VSN:dataset of the source dataset and the complete path and file name of the target file for each MTO operation.

   b) Verify that the MTO target files do not already exist (or can be overwritten).

   c) Vary the MTO volume(s) and channel path(s) offline from the S/390 host.
2. If you will be performing OTM operations:
   a) Make sure that the source files are located on the desired DE volume(s). If you will not be using an existing FCU parameter definition file, write down the complete path and file name of the source file and the VSN:dataset of the target dataset for each OTM operation.

   b) Create and allocate the target datasets. This ensures that the target dataset is registered in the VTOC. Make sure to allocate enough space and to use the appropriate record format and record length for the data to be transferred.

   c) Vary the OTM volume(s) and channel path(s) offline from the S/390 host.

3. If you will be performing OTO operations:

   a) If you will not be using an existing FCU parameter def. file, write down the complete path and file name of the source and target files for each OTM/mto operation.

   b) Use the ALC utility to allocate the intermediate datasets on the OTO volume(s). Make sure to allocate enough space and to use the appropriate record format and record length for the data to be transferred.

   c) Verify that the OTO target files do not already exist (or can be overwritten).

4. Make sure that the desired DE volume definition file (OTO only, or MTO and OTM) is available for use by FCU (**datasetmount.dat** in current directory).

5. Start FCU. If you want to specify any FCU options, start FCU from the DOS prompt. ***Note:*** If you specify the **-nc** option, FCU performs all specified operations continuously, then self-terminates and displays any error information at the DOS prompt.

6. When the FCU main window opens, select the **View-Volume information…** command to open the **Volume information** window, and verify that the desired DE volume(s) is/are available. If not, edit the DE volume definition file as needed.

7. Make sure that the desired FCU parameter definition file is open (displayed in title bar). If not, open the desired FCU parameter definition file using the **File-Open** command. If you want to create a new file using the FCU GUI, see "Creating an FCU parameter definition file using FCU for Windows" (page 123) for instructions.

8. The FCU main window displays the first/next parameter set in the specified FCU parameter definition file. If you want to perform this DE operation, click **Execute**. If not:

   a) You can load the next parameter set using the

   Parameter-Load-Next command .

b) You can delete the current parameter set from the FCU parameter definition file using the **Parameter-Delete** command. The next parameter set loads automatically.

c) You can modify the current parameter set as follows: change the FCU parameters as needed, and then use the **Parameter-Save-Replace** command to replace the previously loaded parameter set with the new parameter set.

d) You can add a new parameter set as follows: change the FCU parameters as needed, and use the **Parameter-Save-Insert** command to insert the new parameter set below the current parameter set.

e) If the **Continuous operation** option is selected, you can open the Error information window before starting the operations to monitor the FCU processing. Move the Error information window so that it does not overlap the FCU main window.

9. When the desired DE operation is displayed, click **Execute** to start the operation. (If the **Execute** button is not enabled, you have not saved the current parameter set.) If the **Continuous operation** option was selected, FCU will process all operations from the current line to the end of the file and then self-terminate. The error information for these operations is placed in the most recent **.log** file(s) in the current directory (e.g., **mto.log**).

10. If you started an MTO operation and the target file already exists, FCU requests overwrite confirmation. Click **OK** to overwrite the target file, or click **Cancel** to cancel the operation.

11. When FCU starts the operation, the Execute window opens and displays the progress of the operation. To cancel the operation in progress, select **Cancel**.

    Note: The Execute dialog window will not appear when the mainframe OS is VSE.

12. When the operation is complete, the Execute window displays the result. If an error occurred, the Error information window opens automatically to display the error. See "Troubleshooting" (page 167) for further information on errors.

13. FCU does not load the next operation automatically. To perform another DE operation, select the desired **Parameter-Load** command, and repeat steps (8) through (12). To exit FCU, select the **File-Exit** command.

Note: After an OTM file transfer from Windows 2000/2003/Windows NT systems, there will be a delay before you can access the DE volume. The length of delay varies according to individual server performance.

Note: After you expand open volumes (LUSE), you will need to reboot Windows 2000/2003/Windows NT systems.

Note: Do not use the open system host to access an DE volume. Use only FAL to access DE volumes. This applies to PC server platforms (e.g., NT) and UNIX-based systems.

Performing DE operations using the non-GUI environment in WIndows

To perform DE file transfer operations for Windows 2000/2003/Windows NT systems in a non-GUI environment:

1. lLog-in as a who has an administrator privilege.
2. Open the command prompt (DOS window) and input the parameters below:

   a) **fcunw** [-cl] [param] ([-v])

   b) **-cl:** Specifying that all the log file for FCU will be cleared before stating FCU.

   c) **param:** This parameter is used as same as (1) in this section for UNIX.

   Note: When this parameter is not specified, the file name of the “Parameter definition file” will be assumed to be “fcudata.prm” and it will attempt to read the detail parameters from the file.

   d) **-v:** This displays the version of **fcunw**.

Note: This parameter cannot be used with any other parameter simultaneously, as it specifies versions.

e)	[Return value]	0 :	Normal end
f)	[Return value]	1 :	Error end

Note: The **fcunw** command requires the “Parameter definition file” to function properly. If there is no “Parameter definition file” or if there is an incorrect parameter in the “Parameter definition file”, the following message will be displayed:

g) [A parameter definition file doesn’t exist, or it is illegal.]

6 Using the File Access Library (FAL)

The File Access Library (FAL) component of DE consists of the object module file **fal.o** (**fal.obj** for Windows 2000/2003/NT systems) and the header file **dataset.h**. The FAL provides several important C functions (Visual C++ for Windows NT/2000/2003) which allow user applications on open system hosts to access S/390 data on the disk array DE volumes. There are two types of FAL, the 32-bit FAL and the 64-bit FAL. The latter is provided for the .

FAL requirements

The FAL functions have the same dataset requirements as FCU (e.g., SAM, standard R0 track format). The FAL also has the following additional requirements:

- The FAL functions support only standard MVS VTOC. The FAL functions cannot access MVS datasets managed by an index VTOC and cannot access VSE datasets when called from user applications.
- The FAL functions are not “thread-safe.” The FAL functions may not operate properly when used by multiple threads within a single process.
- The FAL functions cannot be used by a “signal handler.” If this accidentally happens and memory space is occupied, use **kill** to cancel the processes, and use **ipcrm** to delete the shared memory areas that have KEY=0 (refer to OS manuals). Do not issue the following signals to an DE process (UNIX only):

SIGABRT	SIGIOT	SIGTSTP
SIGALRM	SIGKILL	SIGTTIN
SIGBUS	SIGLWP	SIGTTOU
SIGCANCEL	SIGPOLL	SIGUSR1
SIGCONT	SIGPROF	SIGUSR2
SIGEMT	SIGSEGV	SIGVTALRM
SIGFPE	SIGSTOP	SIGWAITING
SIGFREEZE	SIGSYS	SIGXCPU
SIGILL	SIGTHAW	SIGXFSZ
SIGIO	SIGTRAP	

- The following terminology is reserved for the FAL functions and cannot be used in function names, variable names, or constant symbols in the user application:
    - Words that begin with **dataset** or **fast_**
    - **GetVolSers**

FAL functions

The FAL includes the following C functions (Visual C++ for Windows 2000/2003/NT systems):

- Opening a dataset: **datasetOpen**
- Reading one record from a dataset: **datasetGet**
- Writing one record to a dataset: **datasetPut**
- Closing a dataset: **datasetClose**
- Acquiring error information: **datasetGetLastError**
- Acquiring dataset attribute information:

  datasetGetFileInformation
  datasetFindNextFile
  datasetFindFirstFile
  datasetFindClose
- Converting dataset attribute information:

  datasetGetDsorgString
  datasetGetRecfmString

Opening a dataset

The **datasetOpen** function opens the dataset specified by **pathname** for the type of access specified by **mode**. The table below shows the **datasetOpen** arguments and return values.

datasetHandle = datasetOpen (pathname, mode)

Item	Value	Type	Description
Argument	pathname	char *	VSN:Dataset name VSN = 6-character volser. Volume must be listed in DE volume definition file. Delimiter = : (colon, no spaces allowed) Dataset name: 44 characters max, no spaces allowed.
	mode	char *	r = open dataset for read access w = open dataset for write access
Return value	datasetHandle	DATASET_HANDLE	Handle
	-1		Error end

When the **datasetOpen** function terminates successfully, it returns a handle which identifies the dataset opened. The **datasetHandle** information is used as the argument of subsequent functions such as **datasetGet**, **datasetPut**, or **datasetClose**. Do not change the **datasetHandle** value returned by this function. If the **datasetOpen** function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function.

The **datasetOpen** function has the following restrictions:

- Only one dataset at a time can be open within one process. When multiple datasets need to be opened, the open dataset must be closed before another dataset can be opened. This restriction does not apply to open system files.
- A dataset which is being accessed by the **datasetFindFirstFile** or **datasetFindNextFile** function cannot be opened. The **datasetFindClose** function must be executed before the dataset can be opened. This restriction does not apply to open system files.

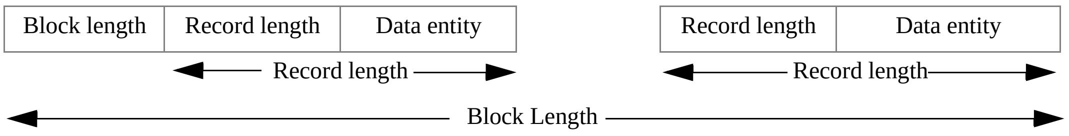
Reading data

The **datasetGet** function reads one record from the specified dataset (**datasetHandle**) and puts the record into a buffer (**buf**) of length **buflen**. The **datasetGet** function extracts only the data entity from each record and does not transfer the BL and RL bytes for variable-length records to the buffer. The table below shows the **datasetGet** arguments and return values.

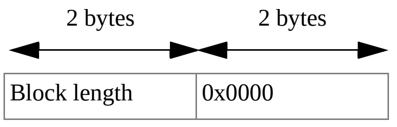
reclen = datasetGet (datasetHandle, buf, buflen)

Item	Value	Type	Description
Argument	datasetHandle	DATASET_HANDLE	The datasetHandle value returned by the **datasetOpen** function.
	buf	char *	Buffer area for storing the read data.
	buflen	long	Size of the buffer area. If the record is larger than **buflen** or equal to zero, **datasetGet** returns an error and does not transfer any data to the **buf**.
Return value	reclen	long	Data entity size transferred to the buffer
	-1		Error end

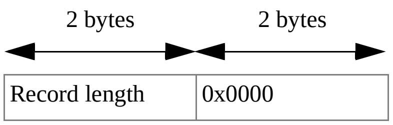
The illustration below shows the format requirements for variable-length records accessed by the **datasetGet** function. Each variable-length block must start with the two-byte BL field, and each variable-length record must start with the two-byte RL field. The **datasetGet** function automatically extracts the data entities without the BL and RL fields.



Block length format:



Record length format:



When the **datasetGet** function terminates successfully, it returns the length of the data entity read from the dataset. If the **datasetGet** function detects the end of dataset (EOF) or terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function. For example, when the **datasetGet** function detects EOF, **datasetGetLastError** will return **DATASET_ERROR_END_OF_FILE**.

Writing data

The **datasetPut** function writes one record from the **buf** into the dataset specified by **datasetHandle**. The table below shows the **datasetPut** arguments and return values.

reclen = datasetPut (datasetHandle, buf, buflen)

Item	Value	Type	Description
Argument	datasetHandle	DATASET_HANDLE	The datasetHandle value returned by the **datasetOpen** function.
	buf	char *	Buffer area for storing the write data.
	buflen	long	Size of the buffer area. If any of the following conditions is detected, **datasetPut** returns an error and does not transfer any data to the dataset: For fixed-length record: **buflen** ≠ RL of the dataset For variable-length record: (**buflen** + 4) > RL of dataset For variable-length record: **buflen** = 0 (no data entity)
Return value	reclen	long	Data entity size written into the dataset.
	-1		Error end

The illustration below shows the format requirements for variable-length records accessed by the **datasetPut** function. When the target dataset is variable-length, the **datasetPut** function takes the data entity from the **buf**, automatically adds the two-byte RL field, and writes the record into the dataset. When the data is written into the dataset, multiple records are blocked within the extent defined by the VTOC of the dataset.

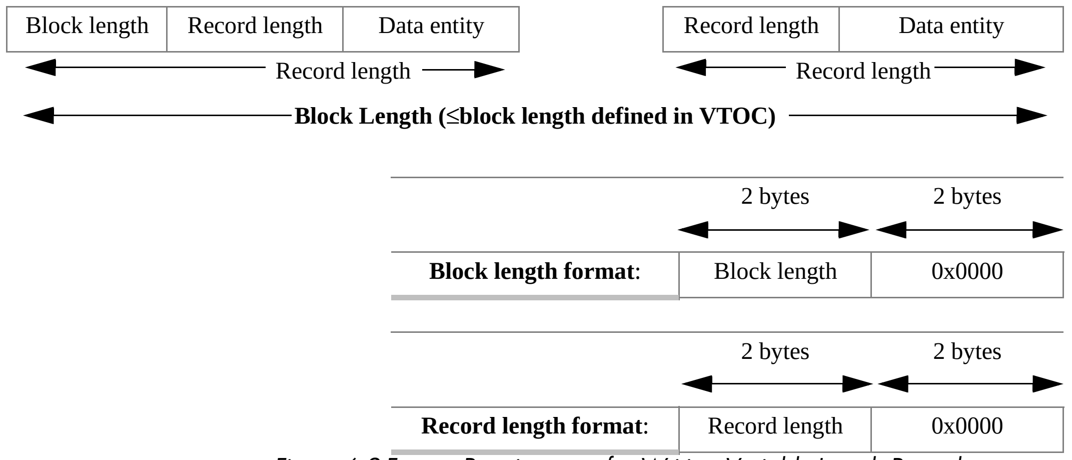


Figure 6.2 Format Requirements for Writing Variable-Length Records

When the **datasetPut** function terminates successfully, it returns the length of the data entity written to the dataset. If the **datasetPut** function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function.

Closing a dataset

The **datasetClose** function closes the dataset specified by **datasetHandle**, which is returned by the **datasetOpen** function. The table below shows the **datasetClose** arguments and return values.

datasetError = datasetClose (datasetHandle)

Item	**Value**	**Type**	**Description**
Argument	datasetHandle	DATASET_HANDLE	The datasetHandle value returned by the **datasetOpen** function.
Return value	0 -1		Normal end Error end

When the **datasetClose** function terminates successfully, it returns a value of 0. If it terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function.

Acquiring error information

The **datasetGetLastError** function acquires the error code information for the most recent error. Errors in FAL functions are defined in **dataset.h**. Errors in UNIX are defined by a standard error file (**errno.h**). Errors in Windows 2000/2003/NT systems are defined by **errno.h** attached with Microsoft Visual C++. The table below shows the **datasetClose** arguments and return values.

datasetLastError = datasetGetLastError()

Item	Value	Type	Description
Argument	none	—	—
Return value	datasetLastError	Long	Error code

Acquiring dataset attributes

FAL provides several functions for acquiring the complete dataset attribute information for one or more datasets: **datasetGetFileInformation**, **datasetFindFirstFile**, **datasetFindNextFile**, and **datasetFindClose**. The dataset attribute information returned by these functions contains:

```
typedef struct DATASET_FIND_DATA {
   unsigned short blockSize; /* Block length  */
   unsigned short recordSize; /* Record length */
   unsigned char dsorg[2]; /* dataset type */
   unsigned char recfm; /* record format */
   char name[44]; /* dataset name */
   unsigned short lastBlockTt; /* last block address (relative 
track number) */
   unsigned char lastBlockR; /* last block address (relative 
record number)*/
}  DATASET_FIND_DATA;
```

Acquiring attribute information for a specific dataset

The **datasetGetFileInformation** function acquires the attribute information for the dataset specified by **pathname** and returns the data into **ffd**. The table below shows the **datasetGetFileInformation** arguments and return values.

datasetError = datasetGetFileInformation (pathname, &ffd)

Item	Value	Type	Description
Argument	pathname	char *	VSN:Dataset name VSN = 6-character volser. Volume must be listed in volume definition file. Delimiter = : (colon, no spaces) Dataset name: 44 characters max, no spaces.
	ffd	DATASET_FIND_DATA	Area where the dataset attribute information is stored.
Return value	0		Normal end
	-1		Error end

When the **DatasetGetFileInformation** function terminates successfully, it returns a value of 0. If it terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function.

The **DatasetGetFileInformation** function has the following restriction:

- The **DatasetGetFileInformation** function cannot be used on an open dataset. Use this function before opening or after closing the dataset.

Acquiring attribute information for multiple datasets

A combination of the **datasetFindFirstFile**, **datasetFindNextFile**, and **datasetFindClose** functions is used to acquire attribute information for more than one dataset in the specified S/390 volume.

1. datasetHandle = datasetFindFirstFile (pathname, &ffd)

   The **datasetFindFirstFile** function acquires the attribute information for the first dataset in the volume specified by **pathname** and returns the data into **ffd**. The table below shows the **datasetFindFirstFile** arguments and return values.

Item	Value	Type	Description
Argument	pathname	char *	VSN VSN = 6-character volser. Volume must be listed in volume definition file.
	ffd	DATASET_FIND_DATA	Area where the dataset attribute information is stored.
Return value	datasetHandle	DATASET_HANDLE	Normal end
	-1		Error end

When the **datasetFindFirstFile** function terminates successfully, it returns a handle which identifies the dataset for which the attribute information was acquired. The **datasetHandle** information is used as the argument of the subsequent functions **datasetFindNextFile** and **datasetFindClose**. Do not change the **datasetHandle** value returned by this function. If the **datasetFindFirstFile** function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function. For example, when the **datasetFindFirstFile** function does not find any datasets in the VTOC, the **datasetGetLastError** function will return **DATASET_ERROR_END_NO_DATASET**.

The **datasetFindFirstFile** function has the following restrictions:

- The **datasetFindFirstFile** function cannot be used on an open dataset. Use this function before opening or after closing the dataset.
- After a dataset has been accessed by the **datasetFindFirstFile** function, the dataset cannot be opened until after the **datasetFindClose** function is called.

2. datasetError = datasetFindNextFile (datasetHandle, &ffd)

   The **datasetFindNextFile** function acquires the attribute information for the next dataset in the volume specified by **datasetHandle** and returns the data into **ffd**. This function can be repeated until “no dataset found” is returned, or until the user application determines that no more information is needed. The table below shows the **datasetFindNextFile** arguments and return values.

Item	Value	Type	Description
Argument	datasetHandle	DATASET_HANDLE	The datasetHandle value returned by the preceding **datasetFindFirstFile** function.
	ffd	DATASET_FIND_DATA	Area where the dataset attribute information is stored.
Return value	0		Normal end
	-1		Error end, or no dataset found

When the **datasetFindNextFile** function terminates successfully, it returns a value of 0. If this function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function. For example, when the **datasetFindNextFile** function cannot find the next dataset in the VTOC, the **datasetGetLastError** function will return **DATASET_ERROR_END_OF_VTOC**.

The **datasetFindNextFile** function has the following restrictions:

- The **datasetFindNextFile** function cannot be used on an open dataset. Use this function before opening or after closing the dataset.
- After a dataset has been accessed by the **datasetFindNextFile** function, the dataset cannot be opened until after the **datasetFindClose** function is called.
- The **datasetFindFirstFile** function must be called prior to **datasetFindNextFile**.

3. datasetError = datasetFindClose (datasetHandle)

The **datasetFindClose** function terminates the acquisition of attribute information by the **datasetFindFirstFile** and **datasetFindNextFile** functions and closes the dataset. The **datasetFindFirstFile** function must be called prior to **datasetFindClose**. The table below shows the **datasetFindClose** arguments and return values.

Item	Value	Type	Description
Argument	datasetHandle	DATASET_HANDLE	The datasetHandle value returned by the preceding **datasetFindFirstFile** function.
Return value	0 -1		Normal end Error end

When the **datasetFindClose** function terminates successfully, it returns a value of 0. If this function terminates unsuccessfully, it returns a value of -1. To get the error code information, execute the **datasetGetLastError** function.

Converting DO and RF information

The FAL provides two functions for converting specific attribute information from a dataset into character strings: **datasetGetDsorgString**, and **datasetGetRecfmString**.

Converting the Dataset Organization (DO) type value

datasetError = datasetGetDsorgString (dsorg, text)

The **datasetGetDsorgString** function converts the dataset organization (DO) type to a three-byte character string. The DO type is specified by **dsorg[2]** in **DATASET_FIND_DATA**. The table below lists the **datasetGetDsorgString** arguments and return values.

Item	Value	Type	Description
Argument	dsorg text	u_char[ ] char [3]	Value of dsorg[2] (two bytes) obtained by the attribute acquisition function **datasetGetFileInformation**, **datasetFindFirstFile**, or **datasetFindNextFile**. Character string indicating the dataset organization (DO) type (3 bytes): PS physical sequential organization VS VSAM organization DA direct access organization PO Partial organization ** Other than above types

Item	Value	Type	Description
Return value	0 -1		Normal end Error end

Converting the Record Format (RF) type value

The **datasetGetRecfmString** function converts the record format (RF) type to a five-byte character string. The RF type is specified by **recfm** in **DATASET_FIND_DATA**. The table below lists the **datasetGetRecfmString** arguments and return values.

datasetError = datasetGetRecfmString (recfm, text)

Item	Value	Type	Description
Argument	recfm text	u_char char [5]	Value of recfm (one byte) obtained by attribute acquisition function **datasetGetFileInformation**, **datasetFindFirstFile**, or **datasetFindNextFile**. Character string (5 bytes) indicating the (RF) type: text[0] F fixed-length record V variable-length record U unknown-length record text[1] B blocking record sp spanned record st standard format record
Return value	0 -1		Normal end Error end

Using the FAL functions

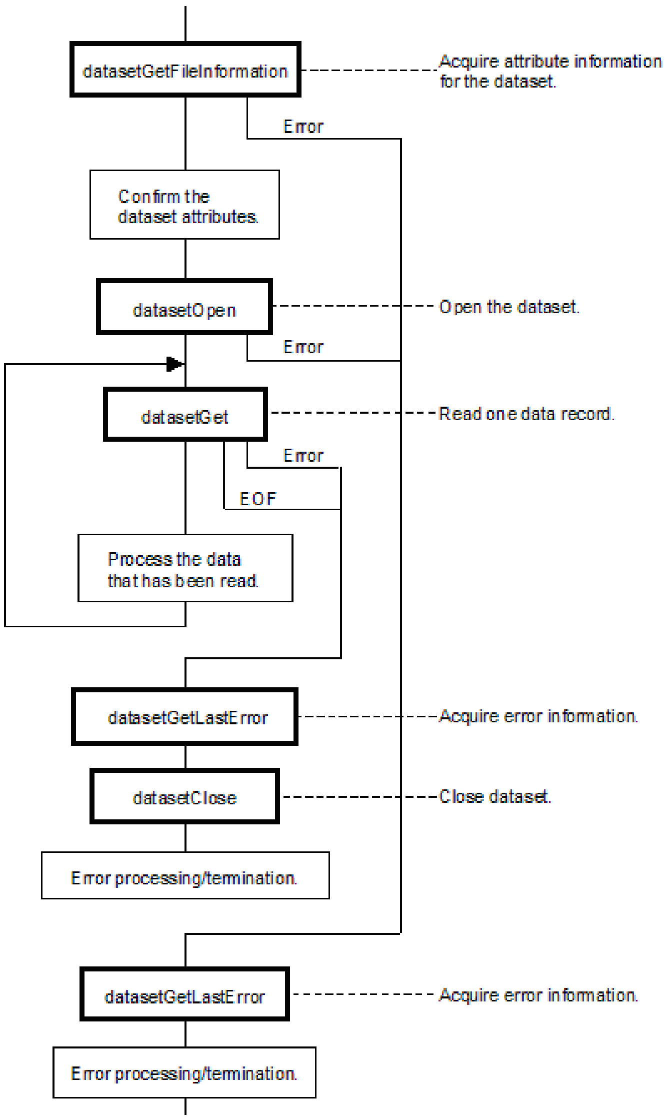
The FAL functions can be executed by any C program on the UNIX host. The FAL does not support C++. The S/390 datasets accessed by the FAL functions must be located on DE volumes. The DE volumes must be installed and configured correctly, the FAL/FCU software must be installed properly, and the DE volume definition file must be available and configured correctly. Since FAL operations do not involve GUI windows, the X windows environment and FcuMf resource file are not required.

The table below shows an example of reading data using the FAL functions. The illustration shows an example of acquiring attribute information using the FAL functions. To use the FAL functions in a C program (Visual C++ for Windows 2000/2003/NT systems):

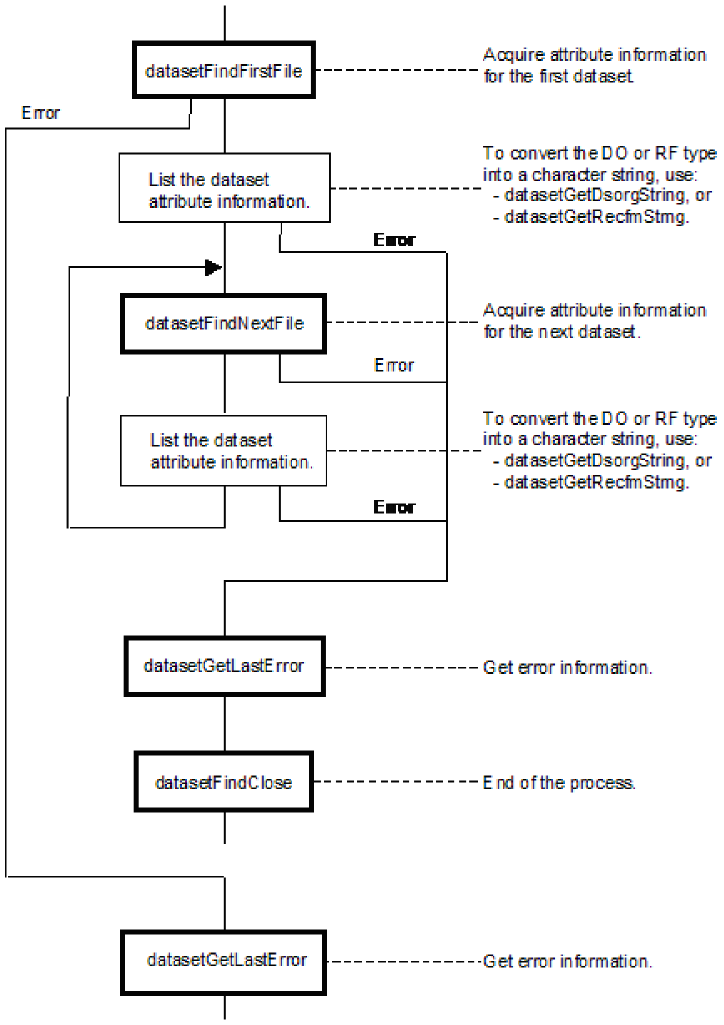
1. Copy the DE volume definition file (**datasetmount.dat**) to the directory containing the C program that will call the FAL C function.
2. Include the FAL header file (**dataset.h**) within the C program that will call the FAL function (e.g., copy **dataset.h** to **/usr/include**).
3. Compile the C program following the instructions for 32-bit FAL or 64-bit FAL for your operating system.:

32-bit FAL	
IBM AIX	# cc -qlanglvl=ansi -o Output file name Source file name /usr/lib/libfal.a **libfal.a** = file name of FAL object module
HP-UX	# cc -Ae +DAportable -o Output file name Source file name /usr/lib/libfal.sl **libfal.sl** = file name of FAL object module
Sun Solaris	# cc -o Output file name Source file name /usr/lib/libfal.so.1 **libfal.so.1** = file name of FAL object module
UNIX /Tru64	# c89 -o Output file name Source file name /usr/lib/libfal.so **libfal.so** = file name of FAL object module

64-bit FAL	
IBM AIX	**# cc -qlanglvl=ansi -q64 -o Output file name Source file name /usr/lib/libfal64.a** **libfal64.a** = file name of FAL object module
HP-UX	**# cc -Ae +DAZ.0W -o Output file name Source file name /usr/lib/pa20_64/libfal64.sl** **libfal64.sl**: = file name of FAL object module.
Solaris™	**# cc xarch=v9 -o Output file name Source file name /usr/lib/sparcv9/libfal64.so.1** **libfal64.so.1**: = file name of FAL object module.
Linux	**# gcc -o Output file name Source file name /usr/lib/libfal.so.1** **libfal.so.1**: This specifies a file name of the object module of the File Access Library.
Windows 2000/2003 Windows NT systems (Visual C++)	Start **Developer Studio** and create a new project. Copy the following three FAL files into the project folder: **dataset.h**, **fal.dll**, **fal.lib** Select **Settings** in the **Projects** menu of **Developer Studio**. In the Project Settings window, select the **Link** tab. Enter **fal.lib** in the **object/library module** field. Build and execute.


datasetGetFileInformation
Acquire attribute information for the dataset.
Error
Confirm the dataset attributes.
datasetOpen
Open the dataset.
Error
datasetGet
Read one data record.
Error
EOF
Process the data that has been read.
datasetGetLastError
Acquire error information.
datasetClose
Close dataset.
Error processing/termination.
datasetGetLastError
Acquire error information.
Error processing/termination.

Example of Reading Data from an S/390 Dataset Using FAL


datasetFindFirstFile
Acquire attribute information for the first dataset
Error
List the dataset attribute information.
To convert the DO or RF type into a character string, use:
- datasetGetDsorgString, or
- datasetGetRecfmStrng.
Error
datasetFindNextFile
Acquire attribute information for the next dataset.
Error
List the dataset attribute information.
To convert the DO or RF type into a character string, use:
- datasetGetDsorgString, or
- datasetGetRecfmStrng.
Error
datasetGetLastError
Get error information.
datasetFindClose
End of the process.
datasetGetLastError
Get error information.

Example of Acquiring S/390 Dataset Attributes Using FAL

Multi-thread function

Specifications

FAL provides the following functions (C programming language):

Information storage area	**dataset_AllocGlobal**
Open a dataset specified by volume name and dataset name	dataset_Open
Read a record specified by dataset	**dataset_Get/dataset_Get2**
Write a record specified by dataset	**dataset_Put/dataset_Put2**
Close a specified dataset	dataset_Close
Free storage area	**dataset_FreeGlobale**
Return a file pointer to top	**dataset_Rewind**
Get a dataset attribute	**dataset_GetFileInformation**, **dataset_FindFirstFile**, **dataset_FindNextFile**, **dataset_FindClose**

Programming restrictions

- You cannot use DE from the Signal Handler.
- The words listed below are reserved words. When the user creates a program using FAL, these words cannot be used for function names, variable names, symbol names, or constant names:
  - dataset
  - fast_
  - GetVolSers
- Do not mix the DE multi-thread function with user API for multi-thread and user API for non-multi-thread.
- This function is only applicable for AIX (32-bit version) and Windows NT/Windows 2000/Windows 2003.

- You do not need a volume definition file when user uses API for multi-thread.
- You can open multiple datasets simultaneously using multi-thread API:
    - **data set_AllocGlobal**: Reserve an area for information of dataset “A”.
    - **dataset_AllocGlobal** : Reserve an area for information of dataset “B”.
    - **datset_Open**: Open dataset “A”.
    - **dataset_Open**: Open dataset “B”.

Information storage area

Format

memError= dataset_AllocGlobal(dgpp,derrno,malloc,free)

Argument	Type	Description
dgpp	void **	Global memory area
derrno	long*	An error information stored area
malloc	void**	malloc()
free	void*	free()
Return value: memError	int	Abnormal end
0		

Note * : When you issue this function, you must issue **dataset_FreeGlobal()** in the end process.

Note** : You must issue this function before **dataset_Open()** and **dataset_FindFirstFile()**.

Arguments

- **dgpp**: Global memory area stored area
- **derrno**: Return an address stored FAL error code
- **malloc**: Specify an address of malloc function. Specify as malloc.
- **free**: Specify an address of free function. Specify as free.

Return value

- When this function ends normally, it returns a **1**.
- When this function ends abnormally, it returns a **0**. For further information, refer to "Troubleshooting" (page 167).

Example

```
void *memptr;  /* global memory area */

long err;        /* global err information */

int retcode;

     :

retcode = dataset_AllocGlobal(&memptr, &err, malloc, free);

     :

retcode = dataset_FreeGlobal(&memptr, &err)
```

Open dataset

Format

datasetError=dataset_Open(global,g_error,devname,dsname,voltype,mode)

Argument	Type	Description
global	void*	Global memory area
g_error	long*	An error information stored area
devname	char*	raw device name
dsname	char*	dataset name
voltypr	char*	volume emulation type
mode	char*	open mod
Return value: datasetError	long	
-1		Abnormal end

Note * : When you issue this function, you must issue **dataset_ Close()** in the end process.

Note** : You must issue this function before **dataset_Open()**, **Get()**, **dataset_Get2()**, **dataset_Put()**, **dataset_Put2()**, **dataset_Rewind(),** and **dataset_GetFileInformation()**.

This function opens a specified dataset (file) with a specified open mode.

Argument

- **global**: Global memory area (Specify a Global memory area gotten by **dataset_AllocGlobal**.)
- **g_error**: Specify an address to store FAL error code.
- **devname**: raw device name (special file)
- **dsname**: dataset name

- **voltype**: Device emulation type (3390-3A/9A/LA,3390-3B/9B/LB, 3380-KA, 3380-KB)
- **mode**:
  - “r”: Read only
  - “w”: Write only
- **Return Value**: When this function ends abnormally, it returns -1.

For further information, see Appendix B.

Example

```
void *memptr;              /* global memory area */

long err,datasetError;      /* global err information */

int retcode;

retcode = dataset_AllocGlobal(&memptr, &err, malloc, free);

 :

datasetError = dataset_Open(memptr, &err,”

 HYPERLINK "\\\\.\\PHYSICALDRIVE1"

\\.\PHYSICALDRIVE1

,”DSN001

, ”3390-3A”,”r”);

      :

datasetError=dataset_Close(memptr, &err);

retcode = dataset_FreeGlobal(&memptr, &err);
```

Read data

Format

reclen = dataset_Get(global, g_error, buf, buflen)

reclen= dataset_Get2(global, g_error, buf, buflen)

Table 1. Table 6.14Arguments, Types and Descriptions for Read Data

Argument	**Type**	**Description**
global	void*	Global memory area
g_error	long*	An error information stored area
buf	char*	Read buffer
buflen	long*	Data length transferred to buffer
Return value: reclen	long*	Data length read to buffer
-1		Abnormal end

Note: This function provides the ability to read a record of a previously opened dataset out to a buffer. Transferred data is real data only.

Argument

- **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
- **g_error:** Specify an address to store FAL error code.
- **buf:** Specify a buffer to store read data.
- **buflen:** Specify buffer size.

Return value

- For **dataset_Get():**
    - When this function ends normally, **reclen** (record length) is returned. (1 record length32760).
    - When this function ends abnormally, "**- 1**" is returned.

    - When this function detects EOF, “**0**” is returned.
- For **dataset_Get2():**
    - When this function ends normally, **reclen** (record length) is returned. (0record length32760)
    - When this function ends abnormally, “**- 1**” is returned.
    - When this function detects EOF, “**DATASET_ERROR_END_OF_FILE** ” is returned.

Note: When “**- 1**” is returned, refer to the content of **g_error** for error code details.

Write data

Format

recren= dataset_ Put(global, g_error, buf, buflen)

recren= dataset_ Put2(global, g_error, buf, buflen)

Argument	Type	Description
global	void*	Global memory area
g_error	long*	An error information stored area
buf	char*	Read buffer
buflen	long*	Data length transferred to buffer
Return value: reclen	long*	Data length read to buffer
-1		Abnormal end

Note*: When buflen is “**0**”, the **dataset_Put** function has ended abnormally, but the **dataset_Put2** function ends normally. (It is possible to handle **0** data.)

Note**: When a full data error occurs, a return value of **dataset_Put2** is “**-1**”, but a return value of **dataset_Put** is “**Y**”. **G_error** is “**- 50**”.

This function writes a record of the previous opened dataset to a buffer. For variable length record formats, this function writes real data to a buffer with record length.

Argument

- **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
- **g_error:** Specify an address to store FAL error code.
- **buf:** Specify a buffer to store write data.
- **buflen:** Specify buffer size.

Return value

When this function ends normally, **reclen** (record length) is returned. When this function ends abnormally, "**- 1**" is returned. When "**- 1**" is returned, refer to the contents of **g_error** for error code details.

Close dataset

Format

datasetError=dataset_Close(global,g_error)

Argument	Type	Description
global	void*	Global memory area
g_error	long*	An error information stored area
Return value: datasetError -1	long	Abnormal end

This function closes a dataset.

Return value

- When this function ends normally, "**0**" is returned.

- When "**- 1**" is returned, refer to the content of **g_error** for error code detail. For further information, see Appendix B.

Free information stored area

Format

memError= dataset_FreeGlobal(dgpp, derrno)

Argument	**Type**	**Description**
dgpp	void*	Global memory area
derrno	long*	An error information stored area
Return value: memError	int	Abnormal end
0		

This function releases information stored area.

Argument

- **dgpp:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
- **derrno:** Specify an address to store FAL error code.

Return value

- When this function ends normally, "**1**" is returned.
- When this function ends abnormally, "**0**" is returned. When "**0**" is returned, refer to the content of **derrno** for error code detail. For further information, see Appendix B.

Initialize target record pointer

Format

datasetError=dataset_Rewind(global,g_error)

Argument	Type	Description
global	void**	Global memory area
g_error	long*	An error information stored area
Return value: datasetError	long	Abnormal end
-1		

Note: When this function is issued before **dataset_Put**, **dataset_Put2**, **dataset_Get,** and **dataset_Get2**, the pointer is returned to the top record. And then next **dataset_Put**, **dataset_Put2**, **dataset_Get,** and **dataset_Get2** are performed from the top record.

Argument

- **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
- **g_error:** Specify an address to store FAL error code.

Return value

- When this function ends normally, "**0**" is returned.
- When this function ends abnormally, "**- 1**" is returned.
- When "**- 1**" is returned, refer to the contents of **g_error** for error code detail. For further information, see Appendix B.

Get dataset attribute information—specified dataset

Format

datasetError= dataset_GetFileInformation(global, g_error, &ffd)

Argument	Type	Description
global	void**	Global memory area
g_error	long*	An error information stored area
ffd	DATASET_FIND_DATA	A dataset attribute information stored area
Return value: datasetError	long	Abnormal end
-1		

An attribute of the opened dataset is returned to **ffd**.

Argument

- **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
- **g_error:** Specify an address to store FAL error code.
- **ffd:** A first dataset attribute information stored area

The dataset attribute information format is shown below:

```
typedef struct DATASET_FIND_DATA {
   unsigned short blockSize;/* Block length    */
   unsigned short recordSize;/* Record length   */
   unsigned char dsorg[2];/* Dataset type    */
   unsigned char recfm;/* Record format   */
   char name[44];        /* Dataset name    */
   unsigned short lastBlockTt;  /* Last block address(relative track number) */
   unsigned char lastBlockR;    /* Last block address(relative block number) */
   unsigned char mftype;     /* Mainframe OS(MVS•VOS3•MSP/VSE/VOS1/XSP) */
}  DATASET_FIND_DATA;
```

Return Value

- When this function ends normally, “**0**” is returned.
- When this function ends abnormally, “**- 1**” is returned.
- When “**- 1**” is returned, refer to the contents of **g_error** for details. For further information, see Appendix B.

Note: * You must issue **dataset_Open()** before this function.

Get dataset attribute information—multiple datasets

Format (1)

datasetHandle=dataset_FindFirstFile(global, g_error, pathname, voltype, and ffd).

Argument	**Type**	**Description**
global	void**	Global memory area
g_error	long*	An error information stored area
ffd	DATASET_FIND_DATA	A dataset attribute information stored area
Return value: datasetError	long	Abnormal end
-1		

This function returns top dataset attribute information specified by raw device name to ffd. This function is used with **dataset_FindFirstFile**, **dataset_FindNextFile** and **dataset_FindClose**.

Argument (1)

- **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)

- **g_error**: Specify an address to store FAL error code.
- **Pathname**: Address of partition name/physical drive name
- **Voltype:** Device emulation type (3390-3A/9A/LA, 3390-3B/9B/LB, 3380-KA, 3380-KB)
- **ffd:** A first dataset attribute information stored area

Return value (1)

- When this function ends normally, "**DATASET_HANDLE**" is returned. This handler is used as an argument for next **dataset_FindNextFile** and **dataset_FindClose** functions.
- When this function ends abnormally, "**- 1**" is returned.
- When "**- 1**" is returned, refer to the contents of **g_error** for error code detail. For further information, see Appendix B.

Note: When there is no dataset in the VTOC, the **g_error** is "**DATASET_ERROR_NO_DATASET**".

Format (2)

datasetError= dataset_FindNextFile(global,g_error,datasetHandle, &ffd)

Argument	Type	Description
global	void*	Global memory area
g_error	long*	An error information stored area
datasetHandle ffd	DATSET_HANDLE DATASET_FIND_DATA	Dataset handler
Return value: datasetError		A dataset attribute information stored area
	long	
-1		Abnormal end

This function gets a second dataset and more attribute information. You can get just the next set of dataset attribute information, or you can use this function until no further dataset information is available or returned.

Argument (2)

- **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
- **g_error:** Specify an address to store FAL error code.
- **datasetHandle:** Specify dataset handler
- **ffd:** Next dataset attribute information stored area

Refer to **dataset_GetFileInformation** for dataset attribute information.

Return value (2)

- When this function ends normally, "**0**" is returned.
- When this function ends abnormally, "**- 1**" is returned.
- When "**- 1**" is returned, refer to the contents of **g_error** and for error code details. For further information, see Appendix B.
- When there is no dataset in the VTOC, the return value is "**- 1**" and error information is **DATASET_ERROR_END_OF_VTOC**.

Note*: You must issue **dataset_FindFirstFile** before this function. When you finish getting dataset attribute information, you must issue **dataset_FindClose** in the end process.

Format (3)

datasetError= dataset_FindClose(global,g_error,datasetHandle))

Argument	Type	Description
global	void*	Global memory area
g_error	long*	An error information stored area
datasetHandle	DATSET_HANDLE	Dataset handler
Return value: datasetError	long	Abnormal end
-1		

This function declares the end of the process, and gets dataset attribute information using **dataset_FindFirstFile** and **dataset_FindNextFile**.

Argument (3)

- **global:** Global memory area (Specify a Global memory area gotten by dataset_AllocGlobal.)
- **g_error:** Specify an address to store FAL error code.
- **datasetHandle:** Specify dataset handler.

Return value (3)

- When this function ends normally, “**0**” is returned.
- When this function ends abnormally, “**- 1**” is returned.
- When “**- 1**” is returned, refer to the content of **g_error** and for error code details.

How to compile

An example of installation including FAL is shown below. For UNIX operating systems, you need to use a C language compiler based on ANSI. You need to include a header file in the program which will be using FAL.

Windows NT/2000/2003 systems

1. Launch Developer Studio.
2. Create a new project.
3. Copy the following FAL files to the project folder/directory:
   - dataset.h
   - falmt.dll
   - falmt.lib
4. Select **SETTING** on the Developer Studio **PROJECT** menu.
5. Select the **LINK** tab in the **Project** setting dialog.
6. Add **falmt.lib** to the **OBJECT/LIBRARY MODULE** column.
7. Build/Execute

AIX systems

```
#cc –qlanglvl=ansi –o output file name source file name /usr/lib/libfalmt.a
```

libfalmt.a:	object module file name of Multi-thread for FAL.

Error information

For details on error messages, see "Troubleshooting" (page 167). The following error codes do not occur for FAL Multi-thread:

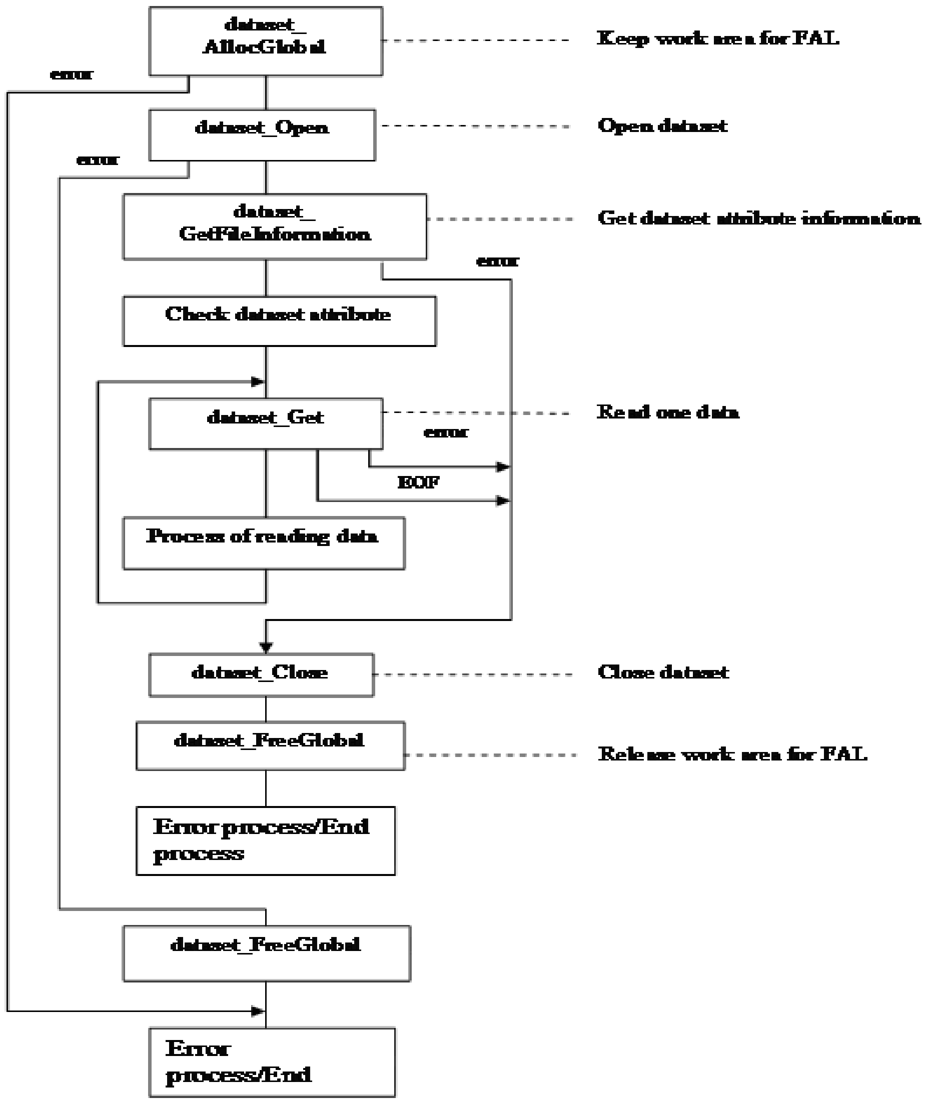
-2, -6, -20, -23, -32

The following error codes **only** occur for FAL Multi-thread:Table 6.23FAL Multi-thread Error Codes

Error Code	Error Message
-29	DATASET_ERROR_CANNOT_MALLOC malloc() function is abnormally ended.
-30	DATASET_ERROR_FREE_INVALID_AREA Invalid area for global area.
-31	DATASET_ERROR_CANNOT_FREE free() function is abnormally ended.

FAL methodology

Example: Read data.



7 Troubleshooting

This section includes resolutions for various error conditions you may encounter.

If you are unable to resolve an error condition, ask your HP support representative for assistance. See "Calling the HP support center" (page 193).

Error conditions

Error Condition	Recommended Action
UNIX files in could not be accessed.	Make sure that the devices have been mounted. If mounting is done during an FCU operation, the results cannot be guaranteed because error information may not be reported to FCU.
Sun Solaris system reports an error indicating **libXm.so.xx is not found**.	Define a path to the Xmlibrary as follows: 1. For C shell, add the following line to the **.cshrc** file in the home directory: **setenv LD_LIBRARY_PATH /usr/dt/lib:$LD_LIBRARY_PATH** 2. F o r non-C shell, add the following two lines to the **.dtprofile** file in the home directory: **LD_LIBRARY_PATH=/usr/dt/lib:$LD_LIBRARY_PATH** **export LD_LIBRARY_PATH**
Windows 2000/2003/NT systems only: FCU reports errors when accessing an FCU parameter definition file.	Remove all space lines from the FCU parameter definition files.
FCU reports code conversion table errors.	If you specified your own code conversion table, make sure that the file name and path are correct. FCU may also report code conversion table errors when the DE volume definition file contains both mainframe and OPEN-*x* DE volumes. Keep the OTO volume definition file separate from the MTO/OTM volume definition file.

Allocater/Formatter error codes

UNIX systems

The Allocator/Formatter codes for UNIX systems are shown below. Codes are classified as F (formatter), A (allocator), or C (common to both).

F/A/C	Error Message	Meaning
A	Allocate check error	An error was found with the allocater check. Check the number of cylinders specified.
A	Allocating dataset failed…	The allocater terminated unsuccessfully.
A	Available volume is not found for allocating	The Volume that device emulation type is OPEN-3/8/9/K doesn’t exist.
F	Available volume is not found for formatting	The Volume (device emulation type OPEN-3/8/9/K) doesn’t exist.
A	Block_length error Block_length=[The range that it can be specified]	The specified block length isn’t correct.
A	Cylinders error	The number of cylinders should be specified in a numerical value or ‘0’.
A	Cylinders large error	The number of cylinders specified has too many characters. It must be specified in fewer than 6 characters.
A	Dataset is full	It exceeds the number of the data set that can be created.
A	Dataset name error (Invalid letter)	Invalid characters are specified or the dataset name isn’t specified.
A	Dataset name error (too long)	A specified dataset name exceeds 44 characters.
C	Emulation type length error in volume definition file	An emulation type parameter in the volume definition file is too long (should be fewer than 11 characters).
F	Format check error	An error was found with the format check. Check the number of primary cylinders specified.
F	Format Failed…	The formatter terminated unsuccessfully.

F/A/C	Error Message	Meaning
C	Partition name length error in volume definition file	A partitions name in the volume definition file is too long.(less than 1025 characters)
F	Primary cylinders error Primary_cylinders =[2-5818]	The number of primary cylinders isn't in the range that can be specified, or it is an invalid letter.
A	Record_format error	The specified record format isn't correct.
A	Record_length error Record_length=[The range that it can be specified]	The specified record length isn't correct.
C	The devname and/or VOLSER-name is not found in volume definition file	Specified the device name or the volume name or those combinations doesn't exist in the file
F	The number of cylinder is too large	There is no cylinder capacity specified.
A	The number of cylinder is too large	There is no cylinder capacity specified.
C	The partition name is invalid in volume definition file	Incorrect partition name is specified in the volume definition file.
A	The VSN of allocating volume is disagreement	VSN on the volume is inconsistent with specified VSN.(or on volume definition file).
A	This dataset is already exists	The same dataset name has already been used.
A	This device is not formatted	A specified device isn't formatted.
F	VOLSER-name error	Incorrect characters are specified in the Volume name.
C	Volume definition file close error	A file close error occurred in the volume definition file. Start again.
C	Volume definition file has no data	Volume definition file has no data
F	Volume definition file is not found	Volume definition file is not found
C	Volume definition file is not valid data	The number of the parameter in the volume definition file is incorrect.
C	Volume definition file read error	A read error occurred in the volume definition file. Start again.
C	Volume definition file record length error	The record length in the parameter definition file is too long (should be fewer than 2081 characters).
C	VSN length error in volume definition file	A VSN in the volume definition file is too long (fewer than 7 characters).

Windows NT systems

The Allocator/Formatter codes for Windows NT systems are shown below. Codes are classified as F (formatter), A (allocator), or C (common to both).

F/A/C	Error message	Meaning
A	Allocating dataset failed.	The allocater terminated unsuccessfully.
A	Allocating dataset failed. Allocate check error	An error was found with the allocater check.Check the number of cylinders specified.
A	Available volume is not found for allocating	The Volume (device emulation type OPEN-3/8/9/K) doesn't exist.
F	Available volume is not found for formatting	The Volume (device emulation type OPEN-3/8/9/K) doesn't exist.
C	Emulation type length error in volume definition file	An emulation type parameter in the volume definition file is too long (needs to be fewer than 11 characters).
F	Format check error	An error was found with the format check. Check the number of primary cylinder specified.
F	Format Failed…	The formatter terminated unsuccessfully.
A	Format information is error	Info on the specified volume is incorrect. Format the volume.
C	Partition name length error in volume definition file	A partition name in the volume definition file is too long (needs to be fewer than 1025 characters).
C	The devname and/or VOLSER-name is not found in volume definition file	The specified device name, or volume name, or combinations of those two, do not exist in the file.
A	The VSN of allocating volume is disagreement	VSN on the volume is inconsistent with specified VSN (or on volume definition file).
A	There is no volume for allocating!	There is no formatted volume.
A	This dataset is already exists	The dataset name already exists.
A	This VOLSER is already exists	The VOLSER name already exists.
C	Volume definition file has no data	Volume definition file has no data.
C	Volume definition file is not found	Volume definition file is not found.
C	Volume definition file read error	A read error occurred in the volume definition file. Start again.
C	Volume definition file record length error	The record length in the parameter definition file is too long (needs to be fewer than 2081 characters).
C	VSN length error in volume definition file	A VSN in the volume definition file is too long (needs to be fewer than 7 characters).

FAL error codes

The error information returned by the **datasetGetLastError** function includes the FAL error information defined in the **dataset.h** file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts.

The FAL error logs for UNIX are **/tmp/fal_error** and **/tmp/fal_error.bak**, and **/tmp/fal_dump** and **/tmp/fal_dump.bak**. The FAL logs for Windows 2000/2003/NT systems are **c:\fal_error** and **c:\fal_error.bak**, and **c:\fal_dump** and **c:\fal_dump.bak**.

Note: Error codes with a negative (-) value are FAL errors. Error codes with a positive value (+) are system errors. UNIX system error codes are defined in the standard error file **errno.h**.

Error Code	Error Message and Description	Recommended Action
-7*	DATASET_ERROR_INVALID_VOLUME The actual VSN and the VSN specified in the DE volume definition file do not match.	Make sure that the VSN in the DE volume definition file is correct.
-8	DATASET_ERROR_DATASET_NOT_FOUND The target dataset was not found.	Make sure that the actual dataset name and the specified dataset name are the same. You can use the MF-File list command in the FCU HELP menu, or VTOC dump data on the S/390 host, to check the dataset name.
-9	DATASET_ERROR_NOT_SUPPORTED The data format is not supported.	Make sure that the dataset was created correctly on the S/390 host.
-10*	DATASET_ERROR_DEVICE_TYPE_NOT_SUPPORTED The device emulation type is not supported.	Make sure that the device emulation type (LVI) is correct in the DE volume definition file. The supported LVIs are 3390-3A, -3B and -3C; and 3380-KA, -KB, and -KC.
-11	DATASET_ERROR_DSORG_NOT_SUPPORTED The dataset organization type is not supported.	Check the DO type using the MF-File list command in the FCU HELP menu, or VTOC dump data on the S/390 host.

Error Code	Error Message and Description	Recommended Action
-12	DATASET_ERROR_RECFM_NOT_SUPPORTED The record format is not supported.	Check the RF type using the MF-File list command in the FCU HELP menu, or VTOC dump data on the S/390 host.
-13*	DATASET_ERROR_INVALID_DATA The data in the VTOC or the dataset is invalid.	Make sure that the VTOC and dataset were created correctly on the S/390 host.
-14*	DATASET_ERROR_VOLUME_DEFINITION_INVALID The format of volume definition file is invalid.	Make sure that the DE volume definition file was created correctly.
-15	DATASET_ERROR_DATASET_NOT_OPENED An attempt was made to read the dataset without opening it.	Make sure that the **datasetOpen** function is called before the **datasetGet** function.
-16	DATASET_ERROR_DATASET_NOT_CLOSED An attempt was made to open the dataset without closing it first.	Make sure the requirements and restrictions are met, for example: Dataset open and close must be used as a pair. More than one dataset cannot be open within one process. **datasetOpen**, **datasetGetFileInformation**, and **datasetFindFirstFile** cannot be used while the dataset is being accessed by **datasetGetFileInformation** or **datasetFindFirstFile**. **datasetGetFileInformation** and **datasetFindFirstFile** cannot be used while the dataset is being accessed **datasetOpen**.
-17	DATASET_ERROR_BUFLEN_SHORT The buffer length specified by **datasetGet** is shorter than the actual record length.	Make sure that the buffer area is larger than the dataset record length.
-18*	DATASET_ERROR_VOLUME_LABEL_INVALID No standard volume label was found, or the contents of the VTOC are illegal.	Make sure that volume initialization is complete and correct on the S/390 host. This error occurs when a system that does not support large files accesses a formatted volume from a system that supports large files. This error also occurs when a data partition size is incorrect for Sun Solaris.
-19*	DATASET_ERROR_VTOC_INVALID No VTOC found, or contents of VTOC are invalid.	Make sure that the VTOC was created correctly on the S/390 host.
-20*	DATASET_ERROR_VOLUME_NOT_DEFINED The specified volume is not defined.	Make sure that the specified volume has been entered correctly in the DE volume definition file.

Error Code	Error Message and Description	Recommended Action
-21	DATASET_ERROR_INVALID_ARGUMENT An argument of the function is invalid.	Make sure that the argument for the FAL function is correct.
-22	DATASET_ERROR_NO_DATASET No dataset was found.	Make sure that the dataset has been created correctly on the S/390 host.
-23*	DATASET_ERROR_NON_STANDARD_R0_EXIST Nonstandard record 0 (R0) exists.	Change the R0 track format to standard track format. FAL cannot write on tracks with nonstandard R0.
-24	DATASET_ERROR_INVALID_MODE The **mode** argument of **datasetOpen** is not valid.	Make sure that the value of the **mode** argument for the **datasetOpen** function is either **r** (for read) or **w** (for write).
-25*	DATASET_ERROR_VOLUME_DEFINED_READ_ONLY The open system host tried to write to a read-only volume.	Make sure that the target dataset for an open system write operation is on a 3390-3A/C or 3380-KA/C volume.
-26	FAL_INTERNAL_ERROR Internal error of FAL	Collect error logs file and error dump file. And make contact a maintenance staff. This error occurs when the open system does not have enough memory.
-27*	DATASET_ERROR_END_OF_VOLUME The end of volume was detected before the end of dataset was detected.	The open system volume/partition size is smaller than the S/390 volume size. Make sure that the partition size is specified correctly on the open system. This error occurs when the open system disk is full or it exceeds a limitation for MTO.
-28	DATASET_ERROR_OVERFLOW Data cannot be written because the dataset is full.	Check the size of the data to be written, and extend the size of the dataset as needed.
-33	DATASET_ERROR_PARAMETER_MISMATCH User-specified RF, BL, RL does not match the RF, BL, RL defined in the VTOC; or RF, BL, RL not specified and not defined in VTOC.	Make sure to specify the correct VSE record option parameters when accessing VSE datasets.
-35	DATASET_ERROR_NO_LICENSE FAL can’t permit execution of software that doesn’t have a software license.	Ensure that the software license is current and correct. If problems persist, contact the HP Support Center.
-36	DATASET_ERROR_TIMEOUT_LICENSE FAL can’t permit execution of software with an expired software license trail time.	Ensure that the trial software license is current and correct. If problems persist, contact the HP Support Center
-37	DATASET_ERROR_HOSTNAME_CHANGE FAL can’t permit execution if the current host and the installed host are not identical and/or the hostname is changed.	Ensure that the current host name has not been changed.

Error Code	Error Message and Description	Recommended Action
-39	DATASET_ERROR_MULTI_VOLUME_DEFINITION_REC ORD_OVER The number of parameter sets for multiple volume definition file exceeded 1000.	Parameter sets more than 1000 cannot be processed. Decrease them not to exceed 1000.
-40	DATASET_ERROR_MULTI_VOLUME_NO_DATASET The data set isn't exist in the next volume.	Check volume serial number in the multiple volume definition file.
-41	DATASET_ERROR_MULTI_VOLUME_NO_TRANSFER	Data cannot be transferred to the dataset that is in middle volume of multiple volume.
-42	DATASET_ERROR_MULTI_VOLUME_DEFINITION_INVA LID_RECORD_LENGTH The record length in the multiple volume definition file is too long.	Specify the record length less than 1400 characters.(not include delimiter).
-43	DATASET_ERROR_MULTI_VOLUME_DEFINITION_PAR AMETER _ERROR The number of volume for one dataset in the multi volume definition file exceeded 31.	Specify the number of volume is less than 31 for one line in the multi volume definition file.
-44	DATASET_ERROR_MULTI_VOLUME_DEFINITION_NO_ DATASET The Dataset name is not specified in the multiple volume definition file.	Specify the dataset name in the head volume information of the multiple volume definition file.
-45	DATASET_ERROR_MULTI_VOLUME_DEFINITION_VSN _LENGTH _ERROR VSN is incorrect in the multiple volume definition file.	Check if VSN length in the multiple definition file is less than 7.
-46	DATASET_ERROR_MULTI_VOLUME_DEFINITION_DSN _LENGTH _ERROR DSN is incorrect in the multiple volume definition file.	Check if DSN length in the multiple definition file is less than 45.
-47	DATASET_ERROR_MULTI_VOLUME_DEFINITION_VOL ID _LENGTH_ERROR The VSN identification length in the multiple volume definition file is too long.	Specify the VSN identification length less than 36 characters.

Error Code	Error Message and Description	Recommended Action
-48	DATASET_ERROR_MULTI_VOLUME_DEFINITION_NO_ NEXT _VOLUME The next VSN is specified in the multiple volume definition file when the volume isn't last on VTOC.	Specified all volumes in the multiple volume definition file.
-50*	DATASET_ERROR_END_OF_FILE End of File (EOF) was detected.	None.
-51*	DATASET_ERROR_END_OF_VTOC End of VTOC was detected.	None.

FCU error codes for UNIX

If FCU for UNIX reports an error, use the **Help-Error** command to view the most recent error. The table below lists and describes the FCU error codes for UNIX and provides instructions for resolving each error condition. The error codes marked by an asterisk (*) may also be reported when I/O access contention for the DE volume occurs between the S/390 and open system hosts. If the cause of the error cannot be identified as described below, check for any illegal I/O contention for the DE volume.

Note: Error codes with a negative value (-) are FCU errors. Error codes with a positive value (+) are system errors. UNIX system error codes are defined in the standard error file **errno.h**.

Note: Note: The error codes with "(C)" in the "Error code" cell are generated only when using with DE Code Converter. For details, see the *DE Code Converter User's Guide*.

Error Code	Error Message and Description	Recommended Action
-100	No parameter file The FCU parameter definition file could not be found.	If you specified the parameter definition file using the [**param**] option, make sure that the specified file exists and the name is correct. If you did not specify the [**param**] option when you started FCU, make sure that the default parameter definition file exists (**fcudata.param** in the current directory).
-101*	Parameter file: Open error An error occurred when opening the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-102*	Parameter file: Read error An error occurred when reading the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-103*	Parameter file: No valid data The parameters in the parameter definition file are not valid.	Make sure that the FCU initiation parameters are entered correctly in the parameter definition file.
-107	Parameter file: CODE_CONV error The code conversion specified in the parameter definition file is not valid.	Make sure that the code conversion is specified as either **EA** or **No**.

Error Code	Error Message and Description	Recommended Action
-108	Parameter file: PADDING error The padding option specified in the parameter definition file is not valid.	Make sure that the padding is specified as either **Yes** or **No**.
-109	Parameter file: DELIMITER error The delimiter option specified in the parameter definition file is not valid.	Make sure that the delimiter is specified as either **CR**, **LF**, or **No**.
-110*	Parameter file: Open error An error occurred when opening and outputting the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-111	Parameter file: Write error An error occurred when writing to the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-112	Parameter file: Close error An error occurred when closing the parameter definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-114	Parameter: No input file name The input file name was not specified.	Make sure to specify the input file name.
-115	Parameter: VSN error The specified VSN is not correct.	Make sure that the specified VSN matches the actual VSN. Make sure that the VSN is separated from the dataset name by a colon (:).
-116	Parameter: Input file name error The specified input file name is not correct.	Make sure that the specified file name matches the actual file name.
-117	Parameter: Dataset name error The specified input dataset name is not correct.	Make sure that the specified dataset name matches the actual dataset name.
-118	Parameter: Output file name error The specified output file name is incorrect.	Make sure that the specified output file name matches the actual output file name.
-119*	Input file: Open error An error occurred when acquiring the dataset attribute information of the input file.	Display the error code using the **Help-Error** command. If an FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual. For example, if a partition name does not match the partition name in the volume definition file, system error code 6 (No such device) is displayed.

Error Code	Error Message and Description	Recommended Action
-120	Overwrite? (OK/Cancel) This message asks you to confirm whether to overwrite the existing file.	The specified open system target file already exists. Select **OK** to overwrite the file, or select **Cancel** to specify a different target file.
-121	Output file: File name error The output file name is not specified.	Make sure that the correct output file name is specified.
-122*	Output file: Open error An OPEN error occurred when checking to see if the output file exists.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-124	Volume definition: MFtype error. Incorrect MFtype is specified in the volume definition file.	Specified MFN or MFA in MFtype of the volume definition file.
-125*	Volume definition: VSN error The VSN specified in the volume definition file is incorrect.	Display the contents of the volume definition file using the **Help-Volume** command. Make sure that the VSN for the specified volume is correct.
-126	Volume definition: Partition name error The partition name specified in the volume definition file is incorrect.	Display the contents of the volume definition file using the **Help-Volume** command. Make sure that the partition name is correct.
-127*	Volume definition: Emulation type error The LVI type specified in the volume definition file is incorrect.	Display the contents of the volume definition file using the **Help-Volume** command. Make sure that the LVI type is correct.
-128*	Volume definition file: Open error An error occurred when opening the volume definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors. For example, if the volume definition file does not yet exist, error code 2 (No such file or directory) is displayed.
-129*	Volume definition file: Read error An error occurred when reading the volume definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-130*	Volume definition file: No data The information found in the volume definition file is not valid.	Display the contents of the volume definition file using the **Help-Volume** command. Make sure that the parameters for each volume are correct.
-131	Volume definition file: Close error An error occurred when closing the volume definition file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-135	Parameter error: No input file name The input VSN is not specified.	Specify the VSN of the S/390 source dataset before selecting the **Help-MF-File** command.
-136	Parameter error: VSN error The input VSN is incorrect.	Make sure that the VSN has six characters.

Error Code	Error Message and Description	Recommended Action
-137	Dataset error: No dataset The specified volume has no datasets.	Make sure that the VSN is correct.
-138*	Dataset error: Search error An error occurred when searching the dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-139	Dataset error: Close error An error occurred when closing the dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-140	Input file error: Invalid organization type The DO type of the dataset is not supported.	Display the attribute information using the **Help-MF-File** command. The DO type must be SAM.
-141	Input file error: Invalid record format The RF type of the dataset is not supported.	Display the attribute information using the **Help-MF-File** command. The RF type must be fixed-length or variable-length.
-142	Input file error: Invalid block length The block length of the dataset is invalid.	Display the attribute information using the **Help-MF-File** command. The block length must be nonzero and cannot be greater than 32 KB.
-143	Input file error: Invalid record length The record length of the dataset is invalid.	Display the attribute information using the **Help-MF-File** command. The record length must be nonzero and cannot be greater than 32 Kobe.
-144*	Input file error: No data No data was found in the specified dataset.	Display the attribute information using the **Help-MF-File** command, and check the dataset size.

Error Code	Error Message and Description	Recommended Action
-150*	Input file: Open error A file open error occurred in the input dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-151*	Output file: Open error A file open error occurred in the output UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-152	Output file: Get file data error A data acquisition error of the output file occurred during an OTM operation.	Collect information such as error log for troubleshooting.
-153	Processing data: Length check error A data length to be processed by OTM does not match.	Make sure that the specified data length matches the actual data length. Collect information such as error log for troubleshooting.
-155	Buffer: Memory allocation error Memory allocation failed.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-160*	Input file: Read error A read error occurred in the input dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-161*	Output file: Write error A write error occurred in the output UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.

Error Code	Error Message and Description	Recommended Action
-162	Output file: Code conversion error An error occurred in the code conversion to the output UNIX file.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172).
-163	Get processing data error The acquisition of processing data failed.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-165	Dataset error: Invalid data An invalid record length was found in the dataset.	Make sure that the S/390 dataset was generated correctly.
-170	Input file: Close error A file close error occurred in the input dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-171	Output file: Close error A file close error occurred in the output UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-180	UNIX/Open system file: Invalid directory name The specified directory name is not valid.	Check the specified directory name.
-181	UNIX file: Not a directory The specified name is not a directory name.	Check the specified directory name.

Error Code	Error Message and Description	Recommended Action
-182*	UNIX/Open system file: Open directory error A directory open error occurred.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-183	UNIX/Open system file: Close directory error A directory close error occurred.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-190	Input file name: No data The input file name is not specified.	The input file name must be specified when you select **File-Save**.
-191	Output file name: No data The output file name is not specified.	The output file name must be specified when you select **File-Save**.
-192	Parameter file name: No data The parameter definition file name is not specified.	The FCU parameter definition file name must be specified when you select **File-Save**.
-200	Parameter file: End line The last parameter set was loaded from the parameter definition file.	The next time you select **File-Load**, the first set of parameters will be loaded.
-201	Parameter file: Direction error The data transfer direction specified in the parameter file is incorrect.	Make sure that the direction (**MTO** or **OTM**) is correct.
-202	Parameter file: Too many data The number of parameter sets for parameter definition file exceeded 100.	The FCU parameter definition file can only store a maximum of 100 parameter sets. If necessary, delete one or more parameter sets to make room for a new parameter set.
-203	Parameter: Empty select error The **Emp** parameter is incorrect.	Make sure that the **Emp=Yes/No** parameter is correct.
-204	Parameter: RDW select error The **RDW** parameter is incorrect.	Make sure that the **RDW=Yes/No** parameter is correct.
-205	RDW error: CODE_CONV not supported Code conversion is not specified as **No** when **RDW=Yes**.	Code conversion cannot be performed when **RDW=Yes**. Change the code conversion parameter to **No**.
-206	RDW error: PADDING not supported Padding is not specified as **No** when **RDW=Yes**.	Padding cannot be processed when **RDW=Yes**. Change the padding parameter to **No**.
-207	RDW error: DELIMITER not supported Delimiter is not specified as **No** when **RDW=Yes**.	Delimiters cannot be processed when **RDW=Yes**. Change the delimiter parameter to **No**.
-210	Parameter file: Comment line This is a comment line in the parameter file.	If you specify **Load**, FCU will move to the next line. You can also replace the comment line with a valid parameter.
-220	Parameter: VSE select error The VSE parameter format is not correct.	Make sure that the number of VSE parameters is correct and that a comma is used correctly to separate the VSE parameters.

Error Code	Error Message and Description	Recommended Action
-221	Parameter: VSE record format error Record format in the VSE parameter is not correct.	Make sure that the record format is set to either one of F/FB/V/VB.
-222	Parameter: VSE record length error Record length in the VSE parameter is not correct.	Make sure that the record length is set to the correct value within the extent allowed.
-223	Parameter: VSE block length error Block length in the VSE parameter is not correct.	Make sure that the block length is set to the correct value within the extent allowed.
-230	No code conv. table file: No code conv. table The code conversion table was not found.	Make sure that the code conversion table file name is correct and that the file exists. This error may also be reported if you mix mainframe and OPEN-x devices in the same DE volume definition file.
-231	Code conv. table: Open error The code conversion table could not be opened.	Refer to the OS user manuals for assistance.
-233	Code conv. table: Close error The code conversion table could not be closed.	Refer to the OS user manuals for assistance.
-234	Code conv. table: Get file data error The size of the code conversion table could not be obtained.	Check the contents of the file. Refer to the OS user manuals for assistance.
-235	Code conv. table: File size error The size of the code conv table is not correct.	Make sure that the size of the code conversion table is 256 bytes.
-236	Code conv. table function: Invalid argument No source data to be converted was found.	Check the contents of the input file, especially the delimiters.
-238	Code conv. table name: No data The file name of the code conversion table is not specified.	If you do not specify **EA** or **No** for the code conversion option, make sure to specify the correct file name of your code conversion table.
-300	Data error: Invalid record length The data length is not correct for the OTM padding function.	Check the source data length and the target record length, and make sure that the record length is correct for the source data entities.
-301	Dataset error: Invalid record format The record format is not correct for the OTM padding function.	For OTM with padding, make sure that the target dataset has fixed-length record format.
-302	Parameter error: Delimiter error The delimiter setting is not correct for the OTM padding function.	If padding=Yes for an OTM operation, the delimiter option must be CR, LF or CRLF.

Error Code	Error Message and Description	Recommended Action
-319*	Dataset: Open error An error occurred when opening the dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual. For example, if the partition name does not match the partition name in the volume definition file, system error code 6 (No such device) is displayed.
-324	O to M error: RDW is not supported	Do not specify the RDW option for OTM operations.
-340	Dataset error: Invalid organization type The DO type of the dataset is not supported.	Display the attribute information using the **Help-MF-File** command. The DO type must be SAM.
-341	Dataset error: Invalid record format The RF of the dataset is not supported.	Display the attribute information using the **Help-MF-File** command. The RF type must be fixed-length or variable-length.
-342	Dataset error: Invalid block length The block length of the dataset is invalid.	Display the attribute information using the **Help-MF-File** command. The block length must be nonzero and cannot be greater than 32 KB.
-343	Input file error: Invalid record length The record length of the dataset is invalid.	Display the attribute information using the **Help-MF-File** command. The record length must be nonzero and cannot be greater than 32 KB.
-350*	Input file: Open error An open error occurred in the input UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-351*	Output file: Open error A file open error occurred in the output dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-352	Input file: Get file data error A data acquisition error for input file occurred during an OTM operation.	Collect information such as error log for trouble shooting.

Error Code	Error Message and Description	Recommended Action
-353	Processing data: Length check error A data length to be processed in OTM operation does not match.	Collect information such as error log for trouble shooting.
-355	Buffer: Memory allocation error Memory allocation failed.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-360*	Input file: Read error A read error occurred in the input UNIX file.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-361*	Output file: Write error A write error occurred in the output dataset.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-362	Output file: Code conversion error An error occurred in the code conversion to the output dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172).
-363	Get processing data error The acquisition of processing data failed.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-370	Input file: Close error A file close error occurred in the input UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.

Error Code	Error Message and Description	Recommended Action
-371	Output file: Close error A file close error occurred in the output dataset.	Display the error code using the **Help-Error** command. If a FAL error code is displayed, refer to "The error information returned by the datasetGetLastError function includes the FAL error information defined in the dataset.h file. The table lists and describes the FAL error codes and provides instructions for resolving each error condition. Error codes marked by an asterisk (*) may also be reported when I/O access contention occurs. If the cause of the error cannot be identified as described in the table, check for illegal I/O access contention for the DE volume between the S/390 and open system hosts." (page 172). If a system error code is displayed, refer to the OS user manual.
-379*	UNIX file: No data No data was found in the input UNIX file.	Make sure to specify an input file which contains data.
-380	No UNIX file The specified UNIX file was not found.	Make sure that the specified UNIX file exists.
-381*	UNIX file: Open error An open error occurred in the UNIX file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-382	Output file: Unsupported record format The record format of the output file is not supported.	Display the attribute information using the **Help-MF-File** command. The RF type must be fixed-length or variable-length.
-383*	Input file: Invalid format The format of the input file is incorrect.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-384*	Input file: Invalid delimiter position The delimiter position in the input file is incorrect. Data record length of input file exceeds that of target dataset, or a record with no data entity is included.	Display the attribute information using the **Help-MF-File** command. Make sure that the record length of the target dataset is correct.
-385	Input file: File seeking error An error occurred when seeking for the input file.	Display the system error code using the **Help-Error** command. Please refer to the OS user manual for information on system errors.
-399	Volume definition: VSN identification length error. The VSN identification length in the volume definition file is too long.	Specify the VSN identification length than 35 characters.
-400	Parameter: Invalid input file name More than one input file name was specified.	Specify only one file name as the input file.

FCU error codes for Windows

If FCU for Windows 2000/2003/NT systems reports an error, use the **View-Error information…** command to view the most recent error. FCU for Windows 2000/2003/NT systems also logs errors in the FCU log file (**fcudata.prm.log**). The table lists and describes the FCU error codes for Windows 2000/2003/NT systems and provides instructions for resolving each error condition.

Note: Error codes with a positive value (+) are system errors. Windows 2000/2003/NT systems system error codes are defined in the **errno.h** file attached with Microsoft Visual C++.

Error Code	Error Message and Description	Recommended Action
-100	Parameter definition file: Open error An error occurred when opening the parameter definition file.	Make sure that the parameter definition file was created correctly. If the parameter definition file was created correctly, check the system error.
-101	Parameter: Count error An error is detected in the parameter count.	Make sure that the parameter count is correct.
-102	Parameter: Direction error The data transfer direction is not correct.	Make sure that the direction is specified correctly as **MTO** or **otm**.
-103	Parameter: Mainframe file name error Mainframe file name is not correct.	Make sure that the mainframe file name is set correctly.
-104	Parameter: Open system file name error Open system file name is not correct.	Make sure that the open system file name is set correctly.
-105	Parameter: Code conversion error Code conversion setting is not correct.	Make sure that the code conversion option is specified as **EA**, **EcA**, **No**, or **File_name** (of your code conversion table). This error may also be reported if you mix 3390/3380 and OPEN-x devices in the same DE volume definition file.
-106	Parameter: Padding error Padding setting is not correct.	Make sure that the padding option is specified as **Yes** or **No**.
-107	Parameter: Delimiter error Delimiter setting is not correct.	Make sure that the delimiter option is specified as **CRLF** or **No**.
-108	Parameter: Add parameter error Delimiter setting is not correct.	If you are adding delimiters for Windows 2000/2003/NT systems, make sure that the delimiter option is specified as **CRLF** (not just **CR** or **LF**).

Error Code	Error Message and Description	Recommended Action
-109	Parameter: Empty duplication error More than one empty setting is specified.	Specify only one empty setting.
-110	Parameter: RDW duplication error More than one RDW setting is specified.	Specify only one RDW setting.
-120	Volume definition file: Open error An error is detected when opening the volume definition file.	Make sure that the volume definition file was created correctly. If the volume definition file is correct, check the system error.
-121	Volume definition file: Length error The record length in the volume definition file is too long.	Specify the record length less than 2080 characters (not including delimiter).
-124	Volume definition: Emulation type Length error An emulation type parameter in the volume definition file is too long.	Specify an emulation type parameter less than 11 characters.
-125	Volume definition: MFtype Length error. Incorrect MFtype is specified in the volume definition file.	Specified MFN or MFA in MFtype of the volume definition file.
-126	Volume definition: VSN identification length error. The VSN identification length in the volume definition file is too long.	Specify the VSN identification length than 35 characters.
-130	Dataset: No dataset error No dataset is found.	Make sure that the mainframe name is specified correctly, or that the dataset is allocated correctly on the specified volume.
-131	Dataset: Search error An error is detected in searching the dataset.	Make sure that the volume definition file name is specified correctly, or that the mainframe file name is specified correctly.
-132	Dataset: Information get error An error is detected in acquiring dataset information.	Make sure that the volume definition file name is specified correctly, or that the mainframe file name is specified correctly.
-133	Dataset: Organization error The specified dataset org. type is not correct.	Make sure that the dataset organization type is specified correctly.
-134	Dataset: Record format error The specified record format is not correct.	Make sure that the record format is specified correctly.
-135	Dataset: Block length error The specified block length is not correct.	Make sure that the block length is specified correctly.
-136	Dataset: Record length error The specified record length is not correct.	Make sure that the record length is specified correctly.
-137	Dataset: Dataset size error The specified dataset size is not correct.	Make sure that the dataset size is specified correctly.

Error Code	Error Message and Description	Recommended Action
-138	Dataset: Close error An error is detected during close operation.	Check the FAL error code and system error code.
-150	Mainframe file: Open error An error is detected when opening the mainframe file.	Check the FAL error code and system error code.
-151	Mainframe file: Read error An error is detected during reading data from the mainframe file.	Check the FAL error code and system error code.
-152	Mainframe file: Write error An error is detected when writing data into the mainframe file.	Check the FAL error code and system error code.
-153	Mainframe file: Close error An error is detected when closing the mainframe file.	Check the FAL error code and system error code.
-154	Mainframe file: Record format error An error is detected in the record format of the mainframe file.	For OTM with the padding function, make sure that the target dataset has fixed-length record format (or change padding to **No**).
-170	Open system file: Open error An error is detected when opening the open system file.	Make sure that the open system file name is specified correctly. Check if any system error is reported.
-171	Open system file: Read error An error is detected when reading data from the open system file.	Check the system error.
-172	Open system file: Write error An error is detected when writing data into the open system file.	Check the system error.
-173	Open system file: Close error An error is detected when closing the open system file.	Check the system error.
-174	Open system file: No data error No dataset is found.	Make sure that the open system file has data. If not, create the appropriate data in the open system file.
-175	Open system file: Delimiter (CR) position error Delimiter (CR) position error is detected. The source data record length exceeds the target record length, or a record with no data entity is included.	Make sure that the open system file name is correct. Make sure that the mainframe dataset name is correct. Make sure that the record length of the open system file is correct.

Error Code	Error Message and Description	Recommended Action
-176	Open system file: Delimiter (LF) position error Delimiter (LF) position error is detected.	Make sure that the open system file name is correct. Make sure that the mainframe dataset name is correct. Make sure that the record length of the open system file is correct.
-177	Open system file: Record format error An illegal record format is found.	Make sure that the open system file name is correct. Make sure that the mainframe dataset name is correct. Make sure the record format (fixed- or variable-length) of the open system file data is correct.
-178	Open system file: Record length error An illegal record length was found. Data length of open system file is too large.	Check the data length of the open system file, and make sure the dataset has the correct record length.
-190	Code conversion error An error was found during code conversion.	Make sure that the dataset size is specified correctly.
-200	Process data get error An error is detected during close operation.	Check the FAL error code and system error code.
-220	External table file: Open error The code conversion table could not be opened.	Check the file name of code conversion table. Check the system error.
-221	External table file: Size error The code conversion table size is not correct.	Make sure that the size is 256 bytes and that the table was created correctly.
-222	External table file: Read error A read error was found when reading the code conversion table.	Check the system error.
-223	External table file: Close error The code conv. table could not be closed.	Check the system error.
-240	Parameter: Direction, PAD, and DEL not matched The combination of **OTM** direction, **PAD=Yes**, and **DEL=No** is not allowed.	For OTM with the padding function, make sure that the delimiter option is specified as **Yes** (or set padding=No).
-241	Parameter: Direction and RDW not matched The combination of **OTM** data transfer direction and **RDW=Yes** is not allowed.	When the DE data transfer direction is **OTM**, make sure that the RDW option is specified as **No**.
-242	Parameter: Code conv. and RDW not matched The combination of **RDW=Yes** and code conversion other than **No** is not allowed.	When the code conversion option is **EA** or **File_name**, make sure that the RDW option is specified as **No**. When RDW=Yes, the code conversion option must be specified as **No**.
-243	Parameter: Padding and RDW not matched The combination of **RDW=Yes** and **padding=Yes** is not allowed.	When the padding option is specified as **Yes**, make sure that the RDW option is specified as **No**. When the RDW option is specified as **Yes**, make sure that the padding option is specified as **No**.

Error Code	Error Message and Description	Recommended Action
-244	Parameter: Delimiter and RDW not matched The combination of **RDW=Yes** and **delimiter=Yes** is not allowed.	When the delimiter option is specified as **Yes**, make sure that the RDW option is specified as **No**. When the RDW option is specified as **Yes**, make sure that the delimiter option is specified as **No**.
-245	Parameter: Specified VOLSER isn't defined Volume Definition file. Specified VOLSER isn't defined the volume definition file.	Check whether specified VOLSER is defined in the volume definition file.
-300	Parameter definition file: Length error The record length in the parameter definition file is too long.	Specify the record length less than 3200 characters (do not include delimiter).
-301	Mainframe file name: Length error An input dataset name(in case of MTO) or an output dataset name(in case of OTM) in the parameter definition file is too long.	Specify an input/output dataset name less than 1025 characters.
-302	Opensystem file name: Length error An input filet name(in case of OTM) or an output file name(in case of MTO) in the parameter definition file is too long.	Specify an input/output file name less than 1025 characters.
-303	Code conversion Length error A code conversion file name in the parameter definition file is too long.	Specify a code conversion file name less than 1025 characters.
-304	VSE : Length error A VSE parameter in the parameter definition file is not corrected.	Specify a VSE parameter less than 21 characters.
-305	VSE record-format: Length error The record format for VSE in the parameter definition file is not corrected.	Specify the record format for VSE less than 3 characters.
-306	VSE record-length: Length error The record length for VSE in the parameter definition file is not corrected.	Specify the record length for VSE less than 6 characters.
-307	VSE block-length: Length error The block length for VSE in the parameter definition file is not corrected.	Specify the block length for VSE less than 6 characters.

Calling the HP support center

If you are unable to resolve an error condition, contact the HP support center for assistance.

Contact Information

In North America, call technical support at 1-800-652-6672, available 24 hours a day, 7 days a week.

Outside North America, call technical support at the nearest location. Telephone numbers for worldwide technical support are listed on the HP website under support:

http://h18006.www1.hp.com/storage/arraysystems.html

Before you call

Be sure to have the following general information available:

- Technical support registration number (if applicable)
- Product serial numbers
- Product model names and numbers
- Operating system type and revision level

Also, be sure to have the following specific information available:

- **Error codes**: FCU error code, FAL error code, SYS error code. Use the FCU GUI to check recent error information (**Help-Error** command for UNIX, **View-Error information** command for NT).
- **FCU parameters**: direction (MTO or OTM), input and output files, and FCU options (code conversion, padding, delimiter, empty file, RDW, VSE record).
- **DE volume definition file**: contents
- **FCU parameter definition file** (if used): contents
- **Command line log** (if possible)

- **FAL error logs.** The FAL logs for UNIX are **/tmp/fal_error** and **/tmp/fal_error.bak**, and **/tmp/fal_dump** and **/tmp/fal_dump.bak**. The FAL logs for Windows 2000/2003/NT systems are **c:\fal_error** and **c:\fal_error.bak**, and **c:\fal_dump** and **c:\fal_dump.bak**.
- **Windows 2000/2003/NT systems only:** FCU log file (e.g., **fcudata.prm.log**), and Dr. Watson's log file (e.g., **c:\WINNT\DRWTSN32.LOG**).
- **Syslog**: error information and other applicable contents

A

Using FCU without the GUI

FCU can be used without the GUI to perform DE operations. To use FCU without the GUI, start FCU using the **-nw** option. The FCU options are:

- The **-nw** option (**nw** = no window) tells FCU to execute the specified DE operation without displaying the GUI. When you use this option (entered as **fcunw** or **fcu -nw**), FCU uses the FCU initiation parameters specified by the **param** option to perform DE operations.

  Note:

- The **-nc** option (**nc** = no checking) tells FCU to execute all specified DE operations without requesting confirmation for FCU parameters or existing MTO target files. If you want to bypass these confirmations, enter **-nc**. FCU will perform the specified operations and overwrite existing MTO target files.
- The **param** option (where param is the FCU parameter definition file) tells FCU whether to use an FCU parameter definition file or a specific FCU initiation parameter set to perform DE operations. The **param** option must have one of the following three values:
  - [blank]. If you want to use the default FCU parameter definition file (**fcudata.param** located in the current directory) to perform DE operations, leave the **param** option blank (do not enter anything).

- **file_name**. If you want to use a different FCU parameter definition file to perform DE operations, enter the filename of the file. Make sure to enter the complete absolute or relative path if the file is not in the current directory.
- **-P + parameters**. If you want to perform one specific DE operation, enter **-P** followed by the FCU initiation parameter set (e.g., **MTO VSN:dataset targetfile No No No**) for the desired DE operation.

a) ***Note:*** FCU for UNIX cannot be used by a "signal handler." If this accidentally happens and memory space is occupied, use **kill** to cancel the processes, and use **ipcrm** to delete the shared memory areas that have KEY=0 (refer to your OS manuals). Do not issue the following signals to an FCU process:

SIGABRT	SIGIOT	SIGTSTP
SIGALRM	SIGKILL	SIGTTIN
SIGBUS	SIGLWP	SIGTTOU
SIGCANCEL	SIGPOLL	SIGUSR1
SIGCONT	SIGPROF	SIGUSR2
SIGEMT	SIGSEGV	SIGVTALRM
SIGFPE	SIGSTOP	SIGWAITING
SIGFREEZE	SIGSYS	SIGXCPU
SIGILL	SIGTHAW	SIGXFSZ
SIGIO	SIGTRAP	

Procedure

To perform DE operations using FCU without the GUI:

1. If you will be using an FCU parameter definition file to perform DE operations, make sure the file contains the correct FCU initiation parameter sets for the DE operations you want to perform. If you will not be using the default FCU parameter definition file, note the name and location of the file.
2. Log in as root on the UNIX server, and enter: **fcunw [-nc] [param]**
   - To perform the DE operations in the default FCU parameter definition file with confirmations, enter: **fcunw**

Example:

```
# fcunw← Start FCU with checking.
File Conversion Utility Ver.01-01-40/00← FCU program version.
mto VSN:dataset file_name EA No LF← First set of parameters.
ok/cancel ? ok← Enter ok or cancel.
Now checking...← Checking for target file.
Complete← Operation completed.

otm file_name VSN:dataset EA No No← Next set of parameters.
ok/cancel ? ok← Enter ok or cancel.
Input file : Open error (-350) ← Error info displayed.
  (Fal error    : xxx
  (System error : xxx

mto VSN:dataset file_name EA No LF← Next set of parameters.
ok/cancel ? ok← Enter ok or cancel.
Now checking...  ← Checking for target file.
OverWrite ? ok/cancel ? ok← Enter ok to overwrite file.
Complete← Operation completed.

mto VSN:dataset file_name EA No LF← Next set of parameters.
ok/cancel ? cancel← Enter ok or cancel.
 :
 :
#
```

   - To perform the DE operations in the default FCU parameter definition file without confirmations, enter: **fcunw -nc**

```
# fcunw -nc ← Start FCU without checking.
File Conversion Utility Ver.01-01-40/00 ← FCU program version.

mto VSN:dataset file_name EA No LF ← First set of parameters.
Now checking... ← Starting DE operation.
Complete ← Operation completed.

otm file_name VSN:dataset EA No No ← Next set of parameters.
Input file : Open error (-350) ← Error info. displayed.
  (Fal error    : xxx
  (System error : xxx

mto VSN:dataset file_name EA No LF ← Next set of parameters.
Now checking... ← Starting DE operation.
Complete ← Operation completed.

mto VSN:dataset file_name EA No LF ← Next set of parameters.
 :
 :
#
```

- To perform the DE operations in a different FCU parameter definition file with confirmations, enter: **fcunw /directory/directory/file_name**
- To perform the DE operations in a different FCU parameter definition file without confirmations, enter: **fcunw -nc /directory/directory/file_name**
- To perform one specific DE operation with confirmations, enter: **fcunw -P mto VSN:dataset targetfile No No No**

```
# fcunw -nc -P mto VSN:dataset file_name EA No LF ← Start FCU without checking.
mto VSN:dataset file_name EA No LF ← Specified FCU parameters.
Now checking... ← Starting DE operation.
Complete ← Operation completed.
#
```

- To perform one specific DE operation without confirmations, enter: **fcunw -nc -P mto VSN:dataset targetfile No No No**

3. If you specified the **-nc** option, FCU will perform all specified DE operations without requesting confirmation for the FCU parameters or for existing MTO target files.
   - If you did not specify the **-nc** option, FCU will display the FCU initiation parameters for the operation to be performed and request

confirmation. Enter **ok** to perform the specified DE operation, or enter **cancel** to load the next set of FCU parameters.

- If you did not specify the **-nc** option and the MTO target file already exists, FCU will request confirmation to overwrite the file. Enter **ok** to overwrite the existing file, or enter **cancel** to load the next set of FCU initiation parameters.

4. When the DE operation starts, FCU displays **Start**. When the operation completes successfully, FCU displays **Complete**. If the operation does not start or complete successfully, FCU displays an error message and loads the next parameter set.
5. When the last FCU initiation parameter set is processed (or canceled), the FCU program terminates and returns an ending status value. The ending status is included in **$status** for C-shell and **$?** for B-shell/K-shell.

   0 = successful completion. All DE operations completed successfully.

   1 = unsuccessful completion. One or more operations did not complete successfully.

Listvol VSN function

The **listvol VSN** function enables FCU users to access the S/390 dataset information without having to launch the FCU GUI (and use the **Help-MF-File** command). The **listvol VSN UNIX** command displays the dataset information for the specified VSN, as shown below. The **listvol VSN** function requires the DE volume definition file.

```
# listvol volser← Specify 6-character VSN.

  Dataset Name        DO      RF     RL      BL      TT     R      EX (Cyl:Trk)
__________________________________________________________________________________
*SAMFILE01.FIX        SAM     F      4096    4096    1      10     100:0
-DAMFILE.F            DAM     F      128     4096    0      10     100:0
?SAMFILE.VSE          SAM     ?      0       0       0      0      0:0

0← Return value (normal end).
#
```

The **listvol VSN** function displays the following information:

- **Dataset name**. An asterisk (*) before the dataset name indicates that DE can process the dataset. A dash (-) indicates that DE cannot process the dataset. A question mark (?) before the dataset name indicates that FCU can process the dataset only if the VSE record option is used to specify the record format (RF), record length (RL), and block length (BL).
- **Dataset organization (DO) type**: SAM, DAM, PAM, VSAM, ??? = unknown. DE supports SAM datasets.
- **Record format (RF)**: F = fixed length, V = variable length, U = undefined length, S = spanned record, ? = unknown. DE supports F and V record formats.
- **Record length (RL)**: in bytes
- **Block length (BL)**: in bytes
- **TT**+**R**: last block address
- **EX (Cyl:Trk)**: data extent size (number of cylinders:number of tracks)
- **Return value**: **0** indicates normal end; **1** indicates error end. If an error occurred, the error code and message are displayed and the error is logged in the error log file.

B

EBCDIC-ASCII code conversion

The table below lists the EBCDIC-ASCII code conversions performed by the default code conversion table provided with FCU.

Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII
00	NUL	NUL	20	DS		40	SP	DS	60	-	ENQ
01	SOH	SOH	21	SOS	a	41			61	/	BEL
02	STX	STX	22	FS	b	42			62		
03	ETX	ETX	23		c	43		s	63		
04	PF		24	BYP	d	44		t	64		
05	HT	RLF	25	LF	SMM	45		u	65		
06	LC	f	26	ETB	IL	46		v	66		
07	DEL	“	27	ESC	CUI	47		w	67		
08	GE	p	28		h	48		x	68		
09	RLF		29		i	49		y	69		
0A	SMM		2A	SW		4A		N	6A		
0B	VT	VT	2B	CUI		4B	.	ACK	6B	,	
0C	FF	FF	2C			4C	<	DC4	6C	%	LF

Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII
0D	CR	CR	2D	ENQ	HT	4D	(		6D	_	~
0E	SO	SO	2E	ACK	LC	4E	+	CUI	6E	>	
0F	SI	SI	2F	BEL	DEL	4F	|	@	6F	?	SUB
10	DLE	DLE	30			50	&	ETB	70		
11	DC1	DC1	31		j	51		z	71		
12	DC2	DC2	32	SYN	BS	52			72		
13	DC3	DC3	33		l	53			73		]
14	TM		34	PN	m	54			74		
15		e	35	RS	n	55		[	75		
16	BS	GE	36	UC	o	56			76		{
17	IL	g	37	EOT	PF	57			77		A
18	CAN	CAN	38		q	58			78		B
19	EM	EM	39		r	59			79	‘	-
1A	CC	k	3A		^	5A	!	SOS	7A	:	
1B	CUI		3B	CU3		5B	$	BYP	7B	#	
1C	IFS	IFS	3C	DC4	TM	5C	*	SW	7C	@	SP
1D	IGS	IGS	3D	NAK		5D	)		7D	‘	ESC
1E	IRS	IRS	3E			5E	:	CU3	7E	=	NAK
1F	IUS	IUS	3F	SUB	CC	5F	~	=	7F	“	FS
80		C	A0		J	C0	{	#	E0	\	*
81	a	/	A1		V	C1	A		E1		
82	b		A2	s		C2	B		E2	S	
83	c		A3	t		C3	C		E3	T	
84	d		A4	u		C4	D		E4	U	
85	e		A5	v		C5	E		E5	V	
86	f		A6	w		C6	F		E6	W	
87	g		A7	x		C7	G		E7	X	

Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII	Hex	EBCDIC	ASCII
88	h		A8	y	‘	C8	H		E8	Y	
89	i		A9	z	:	C9	I		E9	Z	!
8A		D	AA		K	CA		Y	EA		4
8B		E	AB		L	CB		Z	EB		5
8C		F	AC		M	CC			EC		6
8D		G	AD	[	$	CD			ED		7
8E		H	AE		O	CE			EE		8
8F		I	AF		P	CF			EF		9
90			B0		Q	D0	}	‘	F0	0	
91	j		B1		R	D1	J		F1	1	
92	k	,	B2			D2	K	.	F2	2	SYN
93	l	%	B3			D3	L	<	F3	3	
94	m	_	B4			D4	M	(	F4	4	PN
95	n	>	B5			D5	N	+	F5	5	RS
96	o	?	B6			D6	O	|	F6	6	UC
97	p		B7			D7	P	&	F7	7	EOT
98	q		B8		\	D8	Q		F8	8	
99	r		B9			D9	R		F9	9	
9A	^	;	BA		S	DA			FA		
9B			BB		T	DB			FB		
9C			BC		U	DC		0	FC		
9D			BD	]	)	DD		1	FD		
9E			BE		W	DE		2	FE		

Glossary

AL	Arbitrated loop.
ALC	Allocator utility. FMT and ALC utilities are used to format OPEN-*x* logical units (LUNs) and create intermediate datasets for OTO operations.
AL-PA	Arbitrated loop physical address.
BC	HP StorageWorks Business Copy XP. BC lets you maintain up to nine local copies of logical volumes on the disk array.
CA	HP StorageWorks Continuous Access XP. CA lets you create and maintain duplicate copies of local logical volumes on a remote disk array.
Command View	HP StorageWorks Command View XP, a software product for managing XP arrays. Command View runs on a Windows-based management workstation.
command device	An LDEV that transfers RAID Manager commands to BC or CA logical volumes.
CVS	CVS devices (OPEN-x CVS) are custom volumes that are smaller than normal fixed-sized logical disk devices (volumes).
DKC (disk controller unit)	The array cabinet that houses the channel adapters and service processor (SVP).
DKU (disk cabinet unit)	The array cabinets that house the disk array physical disks.

emulation modes	Emulation modes can be assigned to LDEVs to make them operate like standard OPEN system disk drives. The emulation mode of an LDEV determines its capacity. Refer to the appendices for device capacities.
FAL	File Access Library. FAL is a library of C functions (Visual C++ for Windows NT) that provides an application programming interface for data exchange. FAL functions
FC	Fibre Channel.
FC-AL	Fibre Channel arbitrated loop.
FCP	Fibre Channel Protocol.
FCU	File Conversion Utility. FCU provides the GUI and commands for file transfer operations. It includes data exchange options, including EBCDIC-ASCII code conversion and data record padding and delimiters.
FMT	Formatter utility. FMT and ALC utilities are used to format OPEN-*x* logical units (LUNs) and create intermediate datasets for OTO operations.
HBA	Host bus adapter.
HP	Hewlett-Packard Company.
LDEV	Logical device. An LDEV is created when a RAID group is divided into sections using a selected host emulation mode (for example, OPEN-9 or OPEN-M). The number of resulting LDEVs depends on the emulation mode. “LDEV” and “volume” are synonyms.
LUN	Logical unit number. A LUN results from mapping a SCSI logical unit number, port ID, and LDEV ID to a RAID group. The size of the LUN is determined by the emulation mode of the LDEV and the number of LDEVs associated with the LUN. For example, a LUN associated with two OPEN-3 LDEVs has a size of 4,693 MB.
LUSE	Logical Unit Size Expansion, a feature which logically combines LDEVs so they appear as a larger LDEV. This allows a LUN to be associated with 2 to 36 LDEVs. LUSE allows applications to access data requiring large amounts of disk space.

MTO	Mainframe to open system. An operation that transfers data from S/390 (mainframe) datasets to open system files.
OFC	Open Fibre Control.
OPEN-*x*	A general term describing any one of the supported OPEN emulation modes (for example, OPEN-L).
OS	Operating system.
OTM	Open system to mainframe. An operation that transfers data from open system files to S/390 datasets.
OTO	Open system to open system. An operation that transfers data between open systems without connection to a S/390 host.
path	"Path" and "LUN" are synonymous. Paths are created by associating a port, a target, and a LUN ID with one or more LDEVs.
port	A connector on a channel adapter card in the disk array. A port passes data between the disk array and external devices, such as a host server. Ports are named using a port group and port letter, for example, CL1-A.
RAID	Redundant array of independent disks.
remote console PC	The PC running HP StorageWorks Remote Control XP.
Remote Control (RC)	HP StorageWorks Remote Control XP. A software product used for managing XP arrays.
R-SIM	Remote service information message.
SCSI	Small computer system interface.
SIM	Service information message.
SNMP	Simple Network Management Protocol.

SVP	Service processor. A notebook computer built into the disk array. The SVP provides a direct interface to the disk array and is used only by the HP service representative.
TID	Target ID.
VSC	Volume Size Configuration is a feature that defines custom volumes (CVS volumes) that are smaller than normal fixed-sized logical disk devices (volumes).
WWN	World Wide Name. A unique identifier assigned to a Fibre Channel device.

Index